OMA DS Syntax

Approved Version 2.0 – 19 Jul 2011

Open Mobile Alliance

OMA-TS-DS-Syntax-V2_0-20110719-A

Contents

Figures

Tables

1. Scope

This document specifies the XML syntax and semantics used by OMA DS protocols. The DS Syntax protocol is defined by a set of messages that are conveyed between entities participating in a data synchronization operation. The DS Syntax protocol embodies the concept of a SyncML Package. The SyncML Package performs some set of operations. This conceptual "package" permits either a "batch" of multiple operations put together in a single `SyncML` Message or conveyed as separate `SyncML` Messages, each containing a single operation.

Please refer to [DSCONCEPTS] for further information on the OMA DS organization and history.

2. References

2.1 Normative References

[DSCONCEPTS] "Data Synchronization Concepts and Definitions", Open Mobile Alliance™, OMA-TS-DS_Concepts-V2_0,
URL:http://www.openmobilealliance.org/

[DSDEVINF] "OMA DS Device Information", Open Mobile Alliance™, OMA-TS-DS_DevInf-V2_0,
URL:http://www.openmobilealliance.org/

[DSHISTORY] "OMA DS Standards Change History", Open Mobile Alliance™, OMA-WP-SyncML_ChangeHistory,
URL:http://www.openmobilealliance.org/

[DSHTTPBINDING] "SyncML HTTP Binding Specification", Open Mobile Alliance™, OMA-TS-SyncML_HTTPBinding-V1_2_1,
URL:http://www.openmobilealliance.org/

[DSNOTI] "OMA DS Notification", Version 2.0, Open Mobile Alliance™. OMA-TS-DS-Notification-V2_0,
URL:http://www.openmobilealliance.org/

[DSPRO] "Data Synchronization Protocol", Open Mobile Alliance™, OMA-TS-DS_Protocol-V2_0,
URL:http://www.openmobilealliance.org/

[IMEI] "Digital cellular telecommunications system (Phase 2+) (GSM); Universal Mobile Telecommunications System (UMTS); Numbering, addressing and identification" (3G TS 23.003 Version 3.4.0 Release 1999),
URL:http://webapp.etsi.org/action/PU/20000523/ts_123003v030400p.pdf

[IOPPROC] "OMA Interoperability Policy and Process", Open Mobile Alliance™, OMA-IOP-Process-V1_3,
URL:http://www.openmobilealliance.org/

[ISO8601] "Data elements and interchange formats – Information interchange – Representation of dates and times ISO 8601-2000",
URL://www.iso.ch/iso/en/ISOOnline.openerpage

[RFC1321] " The MD5 Message-Digest Algorithm ", R. Rivest, et al., April 1992,
URL:http://www.ietf.org/rfc/rfc1321.txt

[RFC1766] "Tags for the Identification of Languages", H. Alvestrand, March 1995,
URL:http://www.ietf.org/rfc/rfc1766.txt

[RFC2045] "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", N. Freed & N. Borenstein, November 1996,
URL:http://www.ietf.org/rfc/rfc2045.txt

[RFC2119] "Key words for use in RFCs to Indicate Requirement Levels", S. Bradner, March 1997,
URL:http://www.ietf.org/rfc/rfc2119.txt

[RFC2279] "UTF-8, a transformation format of ISO 10646", F. Yergeau, January 1998,
URL:http://www.ietf.org/rfc/rfc2279.txt

[RFC2396] "Uniform Resource Identifiers (URI): Generic Syntax", T. Berners-Lee, et al., August 1998,
URL:http://www.ietf.org/rfc/rfc2396.txt

[WBXML] "WAP Binary XML Content Format Specification", WAP Forum™, WAP-154-WBXML,
URL:http://www.openmobilealliance.org/

[WDP] "Wireless Datagram Protocol Specification", WAP Forum
URL:http://www.openmobilealliance.org/

[WSP] "Wireless Session Protocol specification", WAP Forum
URL:http://www.openmobilealliance.org/

[WTP] "Wireless Transaction Protocol Specification", WAP Forum,

URL:http://www.openmobilealliance.org/

[XML] "Extensible Markup Language (XML) 1.0", World Wide Web Consortium Recommendation, URL:http://www.w3.org/TR/REC-xml

2.2 Informative References

None.

3. Terminology and Conventions

3.1 Conventions

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in RFC2119.

All sections and appendixes, except “Scope” and “Introduction”, are normative, unless they are explicitly indicated to be informative.

Any reference to elements of the Data Synchronization XML Schema is specified in this `typeface`. Any reference to attributes of the Data Synchronization XML Schema is specified in this *`typeface`*.

Schema fragments are shown as:

Content Model:

```
<xs:attribute name="Atomic" type="xs:boolean" default="false"/>
```

Examples are shown as:

Example n: descriptive text

```
<Delete CmdID="401">
  ...
</Delete>
```

Most examples may be validated against the schema by incorporating them into the XML fragments of Appendix B.

Also note that many of the examples manipulate objects relative to the hierarchy of objects shown in Appendix B.

3.2 Definitions

Please refer to the [DSCONCEPTS] document.

3.3 Abbreviations

Please refer to the [DSCONCEPTS] document.

4. Introduction

OMA Data Synchronization (OMA DS) is a specification for a common data synchronization framework and XML-based format, or representation protocol, for synchronizing data on networked devices. OMA Data Synchronization is designed for use between mobile devices that are intermittently connected to the network and network services that are continuously available on the network. OMA Data Synchronization can also be used for peer-to-peer data synchronization. OMA Data Synchronization is specifically designed to handle the case where the network services and the device store the data they are synchronizing in different formats or use different software systems.

This document specifies the XML syntax and semantics used by the OMA Data Synchronization Protocol.

The Data Synchronization Syntax is defined by a set of messages that are conveyed between entities participating in a data synchronization operation. The messages are represented as an XML document. XML is the industry standard for text document mark-up, as defined in [XML].

The Data Synchronization Syntax also can be identified as a MIME content type. MIME is the Internet standard for identifying multipurpose message contents. It provides a useful mechanism for differentiating between different content and document types.

The Data Synchronization Syntax supports protocol models that are based on a request/response command structure, as well as those that are based on a "blind push" command structure.

The Data Synchronization Syntax embodies the concept of a SyncML Package. The SyncML Package performs some set of operations. This conceptual "package" permits either a "batch" of multiple operations put together in a single `SyncML` Message or conveyed as separate `SyncML` Messages, each containing a single operation. `SyncML` Messages are the body of the MIME entities.

4.1 Version History

For a detailed change history of OMA-DS, refer to **[DSHISTORY]**.

Specific Syntax changes include:

4.1.1 Version 1.0.1

`CTCAP` was moved from the Syntax Specification to Device Information [DSDEVINF], and *`FreeID`* and *`FreeMem`* were moved to the Meta Specification.

4.1.2 Version 1.1

`MoreData` and *`NumberOfChanges`* were added. The Specification set was split into Common, DS, and DM.

4.1.3 Version 1.2

`Correlator`, `Field`, `Filter`, *`FilterType`*, `Move`, `Record`, `SourceParent`, and `TargetParent` were added.

4.1.4 Version 2.0

The Meta Specification, including the elements and attributes of `Anchor`, FieldLevel, *`Format`*, *`FreeID`*, *`FreeMem`*, *`Last`*, MaxObjSize, `Meta`, *`Next`*, `NextNonce`, Size, and Type, were incorporated into the Syntax Specification. Support for Fingerprints was added, including *`FP`*, `ID`, and `IDContainer`. The explicit `SyncAlert` element, with its attributes of *`Behaviour`*, *`ChangeLogValidity`*, *`Direction`*, and *`IDValidity`* was created instead of using `Alert`. `SourceClientURI`, `SourceClientParentURI`, `SourceServerURI`, `SourceServerParentURI`, `TargetClientURI`, `TargetClientParentURI`, `TargetServerURI`, and `TargetServerParentURI` were created instead of using `Source`, `SourceParent`, `Target`, `TargetParent` and `LocURI`. `VerDTD` was renamed to *`Version`*. The flags and simple values of *`CmdID`*, *`Code`*, *`NoStatus`*, *`Atomic`*, *`AuthName`*, *`Cmd`*, *`CmdRef`*, *`Correlator`*, *`SftDel`*, *`FieldLevel`*, *`FilterType`*, *`MsgID`*, *`MsgRef`*, *`NextNonce`*, *`NumberOfChanges`*, *`Sequence`*, and *`Version`* were converted from elements to attributes.

5. SyncML

5.1 SyncML Package and Messages

In SyncML, the operations are conceptually bound into a ***SyncML Package***. The SyncML Package is just a conceptual frame for one or more ***SyncML Message***s that are REQUIRED to convey a set of protocol semantics.

A `SyncML` Message is a well-formed XML document and adheres to the Schema, but does not need to be validated. For example, a `SyncML` message does not need to be validated but the XML MUST adhere to whatever explicitly defined order appears in the Schema. The document is identified by the `SyncML` root or document element type. This element type acts as a parent container (i.e., root element type) for the `SyncML` Message.

The `SyncML` Message, as specified before, is an individual XML document. The document consists of a header, specified by the `SyncHdr` element type, and a body, specified by the `SyncBody` element type. The `SyncML` header specifies routing and versioning information about the `SyncML` Message. The `SyncML` body is a container for one or more ***SyncML Command***s. The `SyncML` Commands are specified by individual element types. The `SyncML` Commands act as containers for other element types that describe the specifics of the `SyncML` command, including any data or meta-information.

5.2 SyncML Commands

SyncML defines the following "request" commands:

- `Add`. Allows the originator to ask that a data element or data elements supplied by the originator be added to data accessible to the recipient.
- `Alert`. Allows the originator to notify the recipient. The notification can be used as an application-to-application message or a message intended for display through the recipient's user interface.
- `Copy`. Allows the originator to ask that a data element or data elements accessible to the recipient be copied.
- `Delete`. Allows the originator to ask that a data element or data elements accessible to the recipient be deleted. A `Delete` command can include a request for the archiving of the data.
- `Get`. Allows the originator to ask for a data element or data elements from the recipient. A `Get` can include the resetting of any meta-information that the recipient maintains about the data element or collection.
- `Move`. Allows the originator to ask that a data element or data elements accessible to the recipient to be moved.
- `Put`. Allows the original to put a data element or data elements on to the recipient.
- `Replace`. Allows the originator to ask that a data element or data elements accessible to the recipient be replaced. This command makes a complete replacement of the data element.
- `Sync`. Allows the originator to specify that the included commands be treated as part of the synchronization of two data collections.
- `SyncAlert`. Allows the originator to negotiate the sync type with the recipient and also allows the originator to send fingerprints and data item identifiers to the recipient.

SyncML defines the following "response" commands:

- `Status`. Indicates the completion status of an operation or that an error occurred while processing a previous request, with an optional updated identifier.
- `Results`. Used to return the data results of a `Get` SyncML Command.

The `SyncML` Commands themselves do not fully define the semantics of the SyncML Operation. For example, "Adding" a document to an application to a database may have very different semantics from "Adding" a transaction request to a queue. The semantics of a SyncML Operation are determined by the type of data that is being operated upon. This means that it is possible for an originator to request an operation of a particular recipient that makes no sense to the recipient. In that case, the recipient MUST return an error response status code.

5.3 XML Usage

The SyncML Messages are represented in a mark-up language defined by [XML]. The DS Syntax protocol is an XML application. The DS Syntax Schema defines the XML Schema used to represent a DS Message. The DS Syntax Schema can be found in Section 7, but it is not necessary to read the Schema in order to understand the protocol.

SyncML makes use of XML name spaces. Name spaces MUST be declared on the first element type that uses an element type from the name space.

Names in XML are case sensitive. By convention in the DS Syntax, the element type and attribute list names are specified using the convention that the first character in each word of the name is in upper case text and remainder of the characters in each word of the names specified in lower case text. For example, SyncML for the Sync Mark-up Language tag or *MsgRef* for the Message Reference tag.

The element types in the DS Syntax Schema are defined within a namespace associated with the URI http://www.openmobilealliance.org/tech/profiles/OMA-DS-DS_2_0-Syntax-Schema-V2_0.xsd or the URN urn:oma:xml:ds:syntax:. The DS Syntax Schema are also identified by the ISO 9070 formal public identifier -//SYNCML//Schema SyncML 2.0//EN.

SyncML also makes use of XML standard attributes, such as xml:lang. Any XML standard attribute can be used in a SyncML document.

SyncML Messages are specified using well-formed XML. However, the SyncML Messages need not be valid XML. That is, the SyncML Messages do not need to specify the XML declaration, prolog, or schema locations. They only need to specify the body of the XML document. This restriction allows for the SyncML Messages to be specified with greater terseness than well-formed, valid XML documents.

The following examples should be considered equivalent:

Fully Qualified: This style is both well-formed and valid (provided a copy of the SyncML schema is available). This style is recommended for file formats, such as storing device information for later use.

```
<?xml version="1.0" encoding="UTF-8"?>
<SyncML xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="OMA-SUP-XSD_DS_Syntax_Schema-V2_0.xsd"
    Version="2.0">
  ...

</SyncML>
```

Minimally Qualified: This style uniquely identifies the namespace as being OMA-DS SyncML, and is well-formed, but not valid. This style SHOULD be used when communicating over limited bandwidth.

```
<SyncML xmlns='syncml:syncml2.0' Version="2.0">
  ...
</SyncML>
```

Unqualified: This style MAY be used where the context is clear through other information.

```
<SyncML Version="2.0">
  ...
</SyncML>
```

Namespace Qualified: This style MAY be used where SyncML Information is included inside other XML documents, and the appropriate namespaces are declared. This style MUST be both well-formed and valid. The choice of namespaces (for elements, and attributes) is arbitrary. The use of any style like this SHOULD NOT be used for over-the-air communications.

```
<syncml:SyncML syncmlA:Version="2.0">
  ...
</syncml:SyncML>
```

One of the main advantages of XML is that it is a widely accepted International recommendation for text document mark-up. It provides for both human readability and machine process ability. In addition, XML allows the originator to capture the structure of a document, not just its content. This is extremely useful for applications such as data synchronization, where not just content, but structure semantics is often exchanged.

5.4 WBXML Usage

XML can be viewed as more verbose than alternative binary representations. This is often cited as a reason why it might not be appropriate for low bandwidth network protocols. In most cases, SyncML uses shortened element type and attribute names. This provides a minor reduction in verbosity. Additionally, the `SyncML` Messages can be encoded in a tokenized, binary format defined by [WBXML], or a standard content compression can be applied, such as HTTP's "Content-Encoding: gzip". The use of [WBXML] format or transport layer compression is external to specification of the SyncML protocol and transparent to any SyncML application. The combination of the use of shortened element type names and an alternative binary format makes SyncML competitive, from a compressed format perspective, with alternative, but private, binary representations.

For the purposes of OMA-DS, WBXML 1.1, WBXML 1.2 and WBXML 1.3 are functionally equivalent, and all MUST be accepted in implementations that support WBXML. Effectively, this merely requires the WBXML parser to accept 01, 02 or 03 as the first byte of the document.

6. Mark-up Language Description

The DS Syntax protocol is a document mark-up consisting of XML element and attribute types. This section provides a prose description of this mark-up. The types are defined in terms of their purpose or usage, parent elements, any restrictions on content or use and content model.

Restrictions listed in this document apply to Data Synchronization Protocol. Examples that illustrate the use of each element type can also be found in this document. Examples in this section make use of XML snippets. They are not intended to be complete XML documents, although most can be validated by incorporating them into the XML fragments of Appendix B. They are only provided to illustrate an example usage of the element type in question.

6.1 Static Conformance Requirements by Type

6.1.1 The Common Use Elements and Attributes

The following are common element types and attributes used by numerous other SyncML element types.

The following table further elucidates the static conformance requirements for the SyncML data description elements and attributes for devices conforming to this specification.

Command	Support of Synchronization Server		Support of Synchronization Client	
	Sending	Receiving	Sending	Receiving
Anchor	MUST	MUST	MUST	MUST
Atomic	MAY	MAY	MAY	MAY
AuthName	MAY	MAY	MAY	MAY
Behaviour	MUST	MUST	MUST	MUST
Chal	MUST	MUST	MAY	MUST
ChangeLogValidity	MAY	MUST	MAY	MUST
ClientURI	MUST	MUST	MUST	MUST
Cmd	MUST	MUST	MUST	MUST
CmdID	MUST	MUST	MUST	MUST
CmdRef	MUST	MUST	MUST	MUST
Code	MUST	MUST	MUST	MUST
Correlator	MAY	MAY	MAY	MAY
Cred	MUST	MUST	MUST	MUST
Direction	MUST	MUST	MUST	MUST
Encrypted	MAY	MAY	MAY	MAY
EncryptedKey	MAY	MAY	MAY	MAY
Field	MAY	MAY	MAY	MAY
FieldLevel	MAY	MUST	MAY	MAY

Filter	MAY	MAY	MAY	MAY
FilterType	MAY	MAY	MAY	MAY
Final	MUST	MUST	MUST	MUST
Format	MUST	MUST	MUST	MUST
FP	MAY	MUST	MAY	MAY
FreeID	MAY	MUST	SHOULD	MAY
FreeMem	MAY	MUST	SHOULD	MAY
ID	MAY	MUST	MAY	MAY
IDContainer	MAY	MUST	MAY	MAY
IDValidity	MAY	MUST	MAY	MUST
Last	MUST	MUST	MUST	MUST
MaxMsgSize	MAY	MUST	MAY	MAY
MaxObjSize	MAY	MUST	MAY	MAY
MoreData	MUST	MUST	MAY	MAY
MsgID	MUST	MUST	MUST	MUST
MsgRef	MUST	MUST	MUST	MUST
Next	MUST	MUST	MUST	MUST
NextNonce	MUST	MUST	MUST	MUST
NoStatus	MAY	MUST	MAY	MUST
NumberOfChanges	MAY	MUST	MAY	MAY
Record	MAY	MAY	MAY	MAY
RespURI	MAY	MUST	MAY	MUST
Sequence	MAY	MUST	MAY	MAY
ServerURI	MUST	MUST	MUST	MUST
SessionID	MUST	MUST	MUST	MUST
SftDel	MAY	MAY	MAY	MAY
Size	MAY	MUST	MAY	MUST
SourceClientURI	MUST	MUST	MUST	MUST
SourceClientParentURI	MAY	MAY	MAY	MAY
SourceServerURI	MUST	MUST	MUST	MUST
SourceServerParentURI	MAY	MAY	MAY	MAY

StatusItem	MUST NOT	MUST	MAY	MUST NOT
SyncType	MUST	MUST	MUST	MUST
TargetClientURI	MUST	MUST	MUST	MUST
TargetClientParentURI	MAY	MUST NOT	MUST NOT	MAY
TargetServerURI	MUST	MUST	MUST	MUST
TargetServerParentURI	MAY	MUST NOT	MUST NOT	MAY
Type	MUST	MUST	MUST	MUST
Version	MUST	MUST	MUST	MUST

Table 1: Common Use Elements and Attributes

6.1.2 Message Container Elements

The following element types provide the basic container support for the `SyncML` message.

The following table further elucidates the static conformance requirements for the SyncML data description elements for devices conforming to this specification.

Command	Support of Synchronization Server		Support of Synchronization Client	
	Sending	Receiving	Sending	Receiving
SyncML	MUST	MUST	MUST	MUST
SyncHdr	MUST	MUST	MUST	MUST
SyncBody	MUST	MUST	MUST	MUST

Table 2: Message Container Elements

6.1.3 Data Description Elements

The following element types are used as container elements for data exchanged in a `SyncML` Message.

The following table further elucidates the static conformance requirements for the SyncML data description elements for devices conforming to this specification.

Command	Support of Synchronization Server		Support of Synchronization Client	
	Sending	Receiving	Sending	Receiving
Data	MUST	MUST	MUST	MUST
Item	MUST	MUST	MUST	MUST
Meta	MUST	MUST	MUST	MUST

Table 3: Data Description Elements

6.1.4 Protocol Management Elements

The following table further elucidates the static conformance requirements for the SyncML protocol management elements for devices conforming to this specification.

Command	Support of Synchronization Server		Support of Synchronization Client	
	Sending	Receiving	Sending	Receiving
Status	MUST	MUST	MUST	MUST

Table 4: Protocol Management Elements

6.1.5 Protocol Command Elements

The following table further elucidates the static conformance requirements for the SyncML protocol command elements for devices conforming to this specification.

Command	Support of Synchronization Server		Support of Synchronization Client	
	Sending	Receiving	Sending	Receiving
Add	MUST	MUST	SHOULD	MUST
Alert	MUST	MUST	MUST	MUST
Copy	MAY	MUST / MUST*	MAY	MAY / MUST*
Delete	MUST	MUST	MUST	MUST
Get	MUST	MUST	SHOULD	MUST
Move	MAY	MAY / MUST*	MAY	MAY / MUST*
Put	MUST	MUST	MUST	MUST
Replace	MUST	MUST	MUST	MUST
Results	MUST	MUST	MUST	SHOULD
Sync	MUST	MUST	MUST	MUST
SyncAlert	MUST	MUST	MUST	MUST

Table 5: Protocol Command Elements

* The Copy and Move commands MUST be supported by implementations that support hierarchical sync

6.1.6 Common Types for Elements

The following table further elucidates the common used types for the elements definition.

There are no static conformance requirements for these common types.

Type	Used in Elements
EmptyType	Final, MoreData
CommonOperationType	Add, Copy, Get, Move, Put

Table 6: Common Element Types

6.2 Element, Attribute, and Type Definitions

6.2.1 Add

Usage: Specifies the `SyncML` command to add data to a data collection.

Parent Elements: `Sync`, `SyncBody`

Content Model:

```
<xs:element name="Add" type="CommonOperationType"/>
```

Restrictions:

The `Add` command is generally used to convey to the recipient any additions made to the originator's database. For example, a mobile device will indicate to the network server any additions made to the local calendar database.

The originator of the command SHOULD only send features/properties of the data item that are supported by the recipient. The device information document of the recipient can contain this information.

One or more `Item` element types MUST be specified. The `Item` element type specifies the data item added to the database. The `SourceServerURI` or `SourceClientURI` specified within the `Item` element type uniquely identifies the item, either relative to the corresponding `TargetServerURI` / `TargetClientURI` or `SourceServerURI` / `SourceClientURI` specified in the parent `Sync` command, relative to an absolute URI in the `SourceServerParentURI`, `SourceClientParentURI`, or `TargetClientParentURI` of a hierarchical object whose parent element is `SyncBody`, or by absolute URI with a non-hierarchical object when the parent element is `SyncBody`. Note that only certain combinations of these parameters are valid, as shown in Table 7: Add Source/Target Combinations, Table 8: Add Relative/Absolute URI Combinations, and the examples below.

When synchronizing hierarchical objects, the `Item` element for each object MUST include parent information. For this purpose `SourceServerParentURI`, `SourceClientParentURI`, or `TargetClientParentURI` MUST be used by the sending device referring to an existing parent. When synchronizing hierarchical objects when the parent element is `SyncBody`, these parent elements MUST be absolute URIs, and the `SourceServerURI` or `SourceClientURI` specified within the `Item` element type MUST be relative to the parent element. When the parent element is a `Sync` command all URIs MUST be relative URIs. Refer to Table 7: Add Source/Target Combinations, Table 8: Add Relative/Absolute URI Combinations, and the examples below for valid combinations.

The `Item` elements within an `Add` command MUST NOT attempt to specify a new identifier on the target, that is, specify either `TargetServerURI` or `TargetClientURI`.

When sending an `Add` command to a recipient that does not support Hierarchy, the Parent URI fields MUST NOT be sent, that is, any of `SourceServerParentURI`, `SourceClientParentURI`, `TargetServerParentURI` or `TargetClientParentURI`. When sending an `Add` command to a recipient that does not support Hierarchy when the parent element is `SyncBody`, the `SourceServerURI` or `SourceClientURI` specified within the `Item` element type MUST be absolute URIs. For all other cases, they MUST be relative URIs.

For examples of the use of absolute URIs, see the `Copy` command.

When the client is the recipient, it MAY assign new local identifiers (LUIDs) for the data items specified in this command. However, in such cases the client MUST also notify the server of the new LUID by returning the new LUID in the corresponding `StatusItem` element.

Source/Target Combinations	Without Hierarchy Support		With Hierarchy Support		
	Example 1	Example 2	Example 3	Example 4	Example 5

Sender	Client	Server	Client	Server	Server
SourceServerURI	-	X	-	X	X
SourceClientURI	X	-	X	-	-
TargetServerURI	-	-	-	-	-
TargetClientURI	-	-	-	-	-
SourceServerParentURI	-	-	-	-	X
SourceClientParentURI	-	-	X	-	-
TargetServerParentURI	-	-	-	-	-
TargetClientParentURI	-	-	-	X	-

Table 7: Add Source/Target Combinations

URI Combinations	Without Hierarchy Support		With Hierarchy Support	
Parent Element	*Direct URIs	**Parent URIs	*Direct URIs	**Parent URIs
SyncBody	Absolute	N/A	Relative	Absolute
Sync	Relative	N/A	Relative	Relative

Table 8: Add Relative/Absolute URI Combinations

* Direct URIs refers to SourceServerURI or SourceClientURI.

** Parent URIs refers to SourceServerParentURI, SourceClientParentURI, or TargetClientParentURI.

Status Codes:

If the command completed successfully, then the (201) Item added exception condition is created by the command.

If the recipient determines that the data item already exists on the recipient's database, then the (418) Already exists exception condition is created by the command.

If the originator's authentication credentials specify a principal with insufficient rights to complete the command, then the (401) Unauthorized exception condition is created by the command. If no authentication credentials were specified, then (407) Authentication required exception condition is created by the command. A suitable challenge can also be returned.

Non-specific errors created by the recipient while attempting to complete the command create the (500) Command failed exception condition.

If there is insufficient space on the recipient database for the data item, then the (420) Device full exception condition is created by the command, and the originator SHOULD NOT attempt to add additional data until the recipient has more free space.

If the MIME content type or content format for the data item is not supported by the recipient, then the (415) Unsupported MIME content type or content format exception condition is created by the command.

Example 1: The client requests the server to add 2 new items of the same type, with credentials that provide write access to the current datastore. The *Type* information for all Items' are provided by the Meta element under the Add command. The items are identified by the client LUID, which includes the fingerprint of the item.

```
<Add CmdID="101">
  <Cred>
    <Meta Type="syncml:auth-md5" Format="b64"/>
    <Data>Zz6EivR3yeaaENcRN6lpAQ==</Data>
  </Cred>
  <Meta Format="chr" Type="text/vcard"/>
  <Item>
    <SourceClientURI FP="0123">1001</SourceClientURI>
    <Data>BEGIN:VCARD
VERSION:3.0
...
END:VCARD
    </Data>
  </Item>
  <Item>
    <SourceClientURI FP="1234">1002</SourceClientURI>
    <Data>
VERSION:3.0
...
END:VCARD
    </Data>
  </Item>
</Add>
```

Example 2: The server requests the client to add a new item. The item is identified by the server's GUID. The client may choose to assign a LUID to the item, in which case it would indicate that to the server in the corresponding `StatusItem` element, and the GUID would no longer be used.

```
<Add CmdID="102">
  <Item>
    <SourceServerURI>ABC012345_1003</SourceServerURI>
    <Meta Format="chr" Type="..."/>
    <Data>
      ...
    </Data>
  </Item>
</Add>
```

Example 3: The client requests the server to add a new folder item to the root of the hierarchy.

```
<Add CmdID="103">
  <Item>
    <SourceClientURI FP="2345">1004</SourceClientURI>
    <SourceClientParentURI>./</SourceClientParentURI>
    <Meta Format="chr" Type="application/vnd.omads-folder+xml"/>
    <Data><![CDATA[
      <Folder>
        <name>Work</name>
        <created>20080401T012345</created>
         ...
        <role>Inbox</role>
      </Folder>]]>
    </Data>
  </Item>
</Add>
```

Example 4: The server requests the client to add a new folder item to an existing, mapped folder.

```
<Add CmdID="104">
  <Item>
    <SourceServerURI>ABC012345_1005</SourceServerURI>
    <TargetClientParentURI>1004</TargetClientParentURI>
    <Meta Format="chr" Type="application/vnd.omads-folder+xml"/>
    <Data><![CDATA[
      <Folder>
        <name>OMA</name>
        <created>20080401T123456</created>
         ...
      </Folder>]]>
    </Data>
  </Item>
</Add>
```

Example 5: The server requests the client to add a new item to a folder that is not yet mapped by the client (e.g. the folder was added during this session by the server)

```
<Add CmdID="105">
  <Meta Format="chr" Type="text/x-vcard"/>
  <Item>
    <SourceServerURI>ABC012345_1006</SourceServerURI>
    <SourceServerParentURI>ABC012345_1005</SourceServerParentURI>
    <Data>BEGIN:VCARD
VERSION:3.0
FN:Bruce Smith
N:Smith;Bruce
TEL;WORK;VOICE:+1-919-555-1234
TEL;WORK;FAX:+1-919-555-9876
EMAIL;INTERNET:bruce1@example.com
END:VCARD
    </Data>
  </Item>
</Add>
```

Note: The above examples include all situations that need to be supported if hierarchy is supported. Examples 1 and 2 are the only examples that need to be supported if hierarchy is not supported. No other combinations of `TargetServerURI` / `TargetClientURI`, `SourceServerURI` / `SourceClientURI`, `TargetServerParentURI` / `TargetClientParentURI` or `SourceServerParentURI` / `SourceClientParentURI` should occur. For example, the client should never be in the situation of needing to add a data item to a folder that was added by the server, but is not yet mapped (use `TargetServerParentURI`), since the client could have already sent the new LUID in the corresponding `StatusItem` element when the folder was added before sending its `Add` command. Similarly, the server MUST NOT create LUIDs, and thus should never include a `TargetClientURI` in an `Add` command.

6.2.2 Alert

Usage: Specifies the specific event. The command provides a mechanism for communicating event information to the recipient, such as requesting for next message, last chunk of a large object not received, terminating a session, etc.

Parent Element: `SyncBody`

Content Model:

```
<xs:element name="Alert" type="AlertType"/>
<xs:complexType name="AlertType">
```

```
  <xs:sequence>
    <xs:element ref="Cred" minOccurs="0"/>
    <xs:element ref="Item" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute ref="CmdID" use="required"/>
  <xs:attribute ref="Code" use="required"/>
  <xs:attribute ref="NoStatus" use="optional"/>
</xs:complexType>
```

Restrictions:

The *Code* attribute specifies the status code corresponding to a specific event for the recipient.

Optionally, one or more Item element types MAY be specified. The Item element type specifies parameters for the Alert command. The TargetServerURI / TargetClientURI and SourceServerURI / SourceClientURI specified within the Item element type MUST be an absolute URI.

If the command and the associated action are completed successfully, then the status code '(200) OK' is created by the command.

If the command was accepted successfully, but the Alert action has not yet been executed successfully, then the (202) Accepted for processing exception condition is created by the command. A subsequent exception condition can be created to relate the eventual completion status of the associated Alert action.

Example: The following is an example for a data sync client to request for next message in the case of multiple messages in a package.

```
<Alert CmdID="1" Code="211">
  <Item>
    <TargetServerURI>./Contact/Contacts</TargetServerURI>
    <SourceClientURI>./C\System\Data\Contacts.cdb </SourceClientURI>
  </Item>
</Alert>
```

6.2.3 Anchor

Usage: Specifies the synchronization state information (i.e., sync anchor) for the current and previous synchronization session.

Parent Element: SyncAlert, Status

Content Model:

```
<xs:element name="Anchor">
  <xs:complexType>
    <xs:attribute ref="Last" use="optional"/>
    <xs:attribute ref="Next" use="required"/>
  </xs:complexType>
</xs:element>
<xs:attribute name="Last" type="AnchorType"/>
<xs:attribute name="Next" type="AnchorType"/>
<xs:simpleType name="AnchorType">
  <xs:union memberTypes="xs:dateTime xs:positiveInteger"/>
</xs:simpleType>
```

Restrictions:

The OPTIONAL *Last* attribute specifies the synchronization anchor for the previous synchronization session. The REQUIRED *Next* attribute specifies the synchronization anchor for the current synchronization session.

The value of *Last* and *Next* attributes MUST specify either an UTC based date/time stamp or a monotonically increasing numeric integer. If a date/time stamp, then the text MUST be in the complete representation, basic format defined by ISO8601.

All *Last* or *Next* values sent in a synchronization session by a particular sender MUST be of the same type, and MUST be used such that a comparison operation on values can determine older from newer.

Determination of the ordinal sequence of the version of an existing object in the recipient and the version of the object can be made by comparing the content information of the object with the value on the existing object.

Example:

```
<Anchor Last="2000-08-24T13:30:00Z" Next="2000-08-24T22:13:00Z"/>
```

Note that a comparison operator should find that the value of *Next* is greater than the value of *Last* (if present).

6.2.4 Atomic

Usage: Specifies that the subordinate commands be executed as a set or not at all.

Parent Element: Sync

Content Model:

```
<xs:attribute name="Atomic" type="xs:boolean" default="false"/>
```

Restrictions:

If the command with the *Atomic* attribute set to 'true' completed successfully, then the normal Status for the command should be returned.

If an error occurs while performing a command within a Sync command specified with an *Atomic* attribute, then the (507) Atomic failed exception condition is created by the Sync command. The error status code indicates the failure of the complete command. Separate, individual error status code can also be created that identify specific errors that created the failure.

If a client can execute all the atomic commands together (and thus guarantee the result) then a client MAY split the responses up over multiple messages. If a client cannot execute all the atomic commands together (and thus cannot guarantee the results of commands not executed) and Status responses would go into multiple messages, then the Sync command MUST fail with status code (517) Atomic response too large to fit in message. Previously executed commands within the Sync command MUST be rolled back.

Example:

```
<Sync CmdID="1234" Atomic="true">
  <Add CmdID="1235">
    <Cred>
      <Meta Type="syncml:auth-md5" Format="b64"/>
      <Data>Zz6EivR3yeaaENcRN6lpAQ==</Data>
    </Cred>
    <Item>
      <TargetClientURI>./Foo/pen</TargetClientURI>
      <Data>Yes</Data>
    </Item>
  </Add>
  <Replace CmdID="12346">
    <Cred>
      <Meta Type="syncml:auth-md5" Format="b64"/>
      <Data>Zz6EivR3yeaaENcRN6lpAQ==</Data>
    </Cred>
    <Item>
      <TargetClientURI>./Foo/version</TargetClientURI>
```

```
      <Data>20000401T133000Z</Data>
    </Item>
  </Replace>
</Sync>
```

6.2.5 AuthName

Usage: Specifies the user name for authentication.

Parent Element: `Cred`

Content Model:

```
<xs:attribute name="AuthName" type="xs:string"/>
```

Restrictions:

For Authentication schemes which do not contain an extractable user identifier, the *`AuthName`* attribute is used to hold a user specific identifier.

6.2.6 Behaviour

Usage: Species the behaviour for the current sync session.

Parent Element: `SyncType`

Content Model:

```
<xs:attribute name="Behaviour">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="Preserve"/>
      <xs:enumeration value="Refresh"/>
    </xs:restriction>
  </xs:simpleType>
</xs:attribute>
```

Restrictions:

The *`Behaviour`* attribute specify the behaviour for the current sync session.

The semantics of *`Direction`*, *`Behaviour`*, *`IDValidity`* and *`ChangeLogValidity`* are described in [DSPRO].

6.2.7 Chal

Usage: Specifies an authentication challenge. The recipient of the challenge specifies authentication credentials, of the given authentication scheme and encoding, in the next request.

Parent Elements: `Status`, `SyncHdr`

Content Model:

```
<xs:element name="Chal" type="ChalType"/>
<xs:complexType name="ChalType">
  <xs:sequence>
    <xs:element ref="Meta"/>
    <xs:element ref="NextNonce" minOccurs="0"/>
  </xs:sequence>
```

```
</xs:complexType>
```

Restrictions:

The `Meta` element's *`Type`* and *`Format`* attributes specify the authentication scheme type and credential encoding style, respectively. The optional `NextNonce` element is only used in SHA-256 authentication scheme to specify the new nonce. The next nonce will be used for authentication when the next sync session is started. The next nonce is provided to make sure the nonce is used only once to avoid an easy eavesdropping of the communication

A challenge can be specified against the DS server or datastore. To challenge a DS server, a `Chal` element is sent in the `Status` command corresponding to the `SyncHdr` of the associated `SyncML` request. To challenge a datastore, the `Chal` element is sent in the `Status` command corresponding to the `SyncAlert` or `Sync` command associated with the database.

When the `Chal` element is specified in the `SyncHdr` element, it can be used to update the next nonce to the other side. When the client or server determines to update the next nonce to the other side, the sender can generate the new next nonce and send it in `SyncHdr/Chal/NextNonce` to the receiver, and the receiver updates the old nonce using the received next nonce. The recipient MUST successfully authenticate the credential information in `SyncHdr` before accepting the nonce update. If more than one nonce is transmitted during the session, the latest one MUST be used for the next session.

If absent and if the status code is (200) `Command completed successfully`, then the same credentials SHALL be used in the next `SyncML` request.

If absent and if the status code is (`212`) `Authentication accepted`, then credentials need not be specified for any subsequent `SyncML` requests within the current session. The session is authenticated.

Example: The following is a SHA-256 authentication challenge.

```
<Status CmdID="1" CmdRef="0" Code="407" MsgRef="1" Cmd="SyncHdr">
  <Chal>
    <Meta Format="b64" Type="syncml:auth-sha256"/>
    <NextNonce>Tm9uY2U=</NextNonce>
  </Chal>
</Status>
```

6.2.8 ChangeLogValidity

Usage: Specifies if the change log is valid.

Parent Element: `SyncType`

Content Model:

```
<xs:attribute name="ChangeLogValidity" type="xs:boolean" />
```

Restrictions:

The *`ChangeLogValidity`* attribute specifies if the change log is valid.

The semantics of *`Direction`*, *`Behaviour`*, *`IDValidity`* and *`ChangeLogValidity`* are described in [DSPRO].

6.2.9 ClientURI

Usage: Specifies the `SourceClientURI` or `TargetClientURI` referenced by a `Status` or `Results` element, or the new Client LUID assigned to the associated `ServerURI`.

Parent Elements: `StatusItem`, `Status`, `Results`

Content Model:

```
<xs:element name="ClientURI" type="URIWithFPType"/>
```

Restrictions:

When specified in the `Status` element, specifies the client address specified in the command associated with the response status. When specified in the `StatusItem` element, specifies the client address specified in the specific `Item`, or the new client identifier for the server address specified in the `Item`. When specified in the `Results` element, specifies the client address specified in the associated `Get` command.

The element MUST be specified in a `Status` command corresponding to any `SyncML` command that includes the `SourceClientURI` or `TargetClientURI` element.

Example:

```
<Status CmdID="4321" MsgRef="1" CmdRef="1234" Cmd="Add" Code="200">
  <ServerURI>ABC012345_1012</ServerURI>
  <ClientURI>1012</ClientURI>
</Status>
```

6.2.10 Cmd

Usage: Specifies the name of the SyncML command referenced by a `Status` element.

Parent Element: `Status`

Content Model:

```
<xs:attribute name="Cmd">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="Add"/>
      <xs:enumeration value="Alert"/>
      <xs:enumeration value="Copy"/>
      <xs:enumeration value="Delete"/>
      <xs:enumeration value="Get"/>
      <xs:enumeration value="Move"/>
      <xs:enumeration value="Put"/>
      <xs:enumeration value="Replace"/>
      <xs:enumeration value="Results"/>
      <xs:enumeration value="Status"/>
      <xs:enumeration value="Sync"/>
      <xs:enumeration value="SyncAlert"/>
      <xs:enumeration value="SyncHdr"/>
    </xs:restriction>
  </xs:simpleType>
</xs:attribute>
```

Restrictions:

The value MUST be one of "`Add`", "`Alert`", "`Copy`", "`Delete`", "`Get`", "`Move`", "`Put`", "`Replace`", "`Results`", "`Status`", "`Sync`", "`SyncAlert`" when the *`CmdRef`* attribute has a value greater than "0"or be "`SyncHdr`" when the *`CmdRef`* attribute has a value of "0".

Example:

```
<Status CmdID="4321" MsgRef="1" CmdRef="1234" Cmd="Add" >
  ...
</Status>
```

6.2.11 CmdID

Usage: Specifies a SyncML message-unique command identifier. This is a positive integer used to uniquely reference the Protocol Management Elements [6.1.4] or Protocol Command Elements [6.1.5] in a message. Generally this is referenced in the *CmdRef* attribute of a Results or Status element.

Parent Elements: Add, Alert, Copy, Delete, Get, Move, Put, Replace, Results, Status, Sync, SyncAlert

Content Model:

```
<xs:attribute name="CmdID" type="xs:positiveInteger"/>
```

Restrictions:

The value MUST be unique within each SyncML Message.

The attribute MUST be present on each of the Protocol Management Elements [6.1.4] or Protocol Command Elements [6.1.5] in a message.

Example:

```
<Add CmdID="1234" >
  <Cred>
    <Meta Type="syncml:auth-md5" Format="b64"/>
    <Data>Zz6EivR3yeaaENcRN6lpAQ==</Data>
  </Cred>
  <Item>
    <SourceClientURI>./12</SourceClientURI>
    <Meta Type="text/directory;profile=vCard"/>
    <Data>BEGIN:VCARD
VERSION:3.0
FN:Smith;Bruce
N:Bruce Smith
TEL;TYPE=WORK;VOICE:+1-919-555-1234
END:VCARD
    </Data>
  </Item>
</Add>
```

6.2.12 CmdRef

Usage: Specifies the *CmdID* referenced by a Results or Status element.

Parent Elements: Results, Status

Content Model:

```
<xs:attribute name="CmdRef" type="xs:nonNegativeInteger"/>
```

Restrictions:

The *CmdRef* attribute MUST refer to the identifier of the SyncML command referenced by the Results or Status element.

The value "0" refers to the SyncHdr of the corresponding message.

Example:

```
<Status CmdID="4321" MsgRef="1" CmdRef="1234" Cmd="Add" Code="401">
  <Chal>
```

```
    <Meta Format="b64" Type="syncml:auth-md5"/>
    <NextNonce>ZG9iZWhhdmUNCg==</NextNonce>
  </Chal>
</Status>
```

6.2.13 Code

Usage: Specifies a status code or alert code.

Parent Elements: `Alert`, `Status`

Content Model:

```
<xs:attribute name="Code" type="xs:positiveInteger"/>
```

Restrictions:

This attribute is used to specify a status code or alert code.

Example:

```
<Alert CmdID="1234" Code="222"/>
```

6.2.14 CommonOperationType

Usage: Specifies the complex type for the `SyncML` commands which have the common operation type.

Parent Elements: `Add`, `Copy`, `Get`, `Move`, `Put`

Content Model:

```
<xs:complexType name="ComplexOperationType"/>
  <xs:sequence>
    <xs:element ref="Cred" minOccurs="0"/>
    <xs:element ref="Meta" minOccurs="0"/>
    <xs:element ref="Item" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute ref="CmdID" use="required"/>
  <xs:attribute ref="NoStatus" use="optional"/>
</xs:complexType>
```

Example:

```
<xs:element name="Add" type="CommonOperationType" />
```

6.2.15 Copy

Usage: Specifies the `SyncML` command to copy data items from one location (data store and/or folder) to another in the recipient's database.

Parent Elements: `Sync`, `SyncBody`

Content Model:

```
<xs:element name="Copy" type="CommonOperationType"/>
```

Restrictions:

It is implementation dependent whether a physical copy of the item is made in the recipient, or whether a shortcut or pointer is created to the source item in the target location.

The Copy command in this version of the specification is NOT intended to be used to attempt to change the MIME content type of a data item, compress the data item or otherwise transform a target data item. Both the source and target locations MUST support the MIME content type of the data item. The result of any differences between the supported field lists of different datastores is undefined.

One or more Item element types MUST be specified. The Item element type specifies the data item to be copied on the recipient's database. Any Data specified in Item SHOULD be ignored by the recipient. The source data item, and the source and target parent datastores and folders MUST exist.

If specified within a Sync element type to reference items within the current datastore, a TargetServerURI, TargetClientURI, SourceServerURI, or SourceClientURI specified within the Item element type in the Copy command SHOULD be a relative URI, as relative to the corresponding TargetServerURI / TargetClientURI or SourceServerURI / SourceClientURI specified in the parent Sync command.

If specified within a Sync element type to reference items outside the current datastore, a TargetServerURI, TargetClientURI, SourceServerURI or SourceClientURI specified within the Item element type in the Copy command SHOULD be a relative URI, as relative to the corresponding TargetClientParentURI or SourceClientParentURI, which SHOULD be an absolute URI.

If specified within a SyncBody element type, a TargetServerURI, TargetClientURI, or SourceClientURI specified within the Item element type in the Copy command SHOULD be a relative URI, as relative to the corresponding TargetClientParentURI or SourceClientParentURI, which SHOULD be an absolute URI.

When specified by the client, Item elements of Copy commands MUST only contain client identifiers (SourceClientURI, SourceClientParentURI, TargetClientURI, and TargetClientParentURI). Note the implication that any item created by the server must be mapped by the client before it can be referenced by the client.

When specified by the server, Item elements of Copy commands MUST contain a TargetServerURI and MUST NOT contain a TargetClientURI. TargetServerParentURI and SourceServerParentURI SHOULD only be used within a Sync command, and SHOULD be relative URIs, as relative to the corresponding TargetServerURI / TargetClientURI or SourceServerURI / SourceClientURI specified in the parent Sync command. Note the implication that servers cannot Copy to or from folders outside the current datastore that have not been mapped by the client. This is to reduce complexity, by avoiding ambiguous parent addresses of the form "./LUID/LUID/GUID/GUID".

The Item element for each object MUST include parent information for both the source and the target location.

When the client is the recipient, it MAY assign new local identifiers (LUIDs) for the data items specified in this command. However, in such cases the client MUST also notify the server of the item identifier correlation by returning the new LUID in the corresponding StatusItem element.

Source/Target Combinations	**Valid inside Sync or SyncBody**		**Only Valid in Sync**				
	Valid within single or multiple datastores			**Only Valid within the current datastore (relative Parent URIs)**			
	Example 1/2	**Example 3/4**	**Example 5/6**	**Example 7**	**Example 8**	**Example 9**	**Example 10**
Sender	**Client**	**Server**	**Server**	**Server**	**Server**	**Server**	**Server**
Use Relative URIs for object identifiers.							
SourceServerURI	-	-	X	-	X	X	X

SourceClientURI	X	X	-	X	-	-	-
TargetServerURI	-	X	X	X	X	X	X
TargetClientURI	X	-	-	-	-	-	-
When Inside Sync, use Relative URIs for Parent Identifiers within the same datastore. When Inside SyncBody, or to reference other datastores, use Absolute URIs for the Parent Identifiers.							
SourceServerParentURI (Relative URI only)	-	-	-	-	-	X	X
SourceClientParentURI	X	X	X	X	X	-	-
TargetServerParentURI (Relative URI only)	-	-	-	X	X	-	X
TargetClientParentURI	X	X	X	-	-	X	-

Table 9: Copy Source/Target Combinations

Status Codes:

If the command completed successfully, then the (201) `Item added` exception condition is created by the command.

If the originator's authentication credentials specify a principal with insufficient rights to complete the command, then the (401) `Unauthorized` exception condition is created by the command. If no authentication credentials were specified, then (407) `Authentication required` exception condition is created by the command. A suitable challenge can also be returned.

If the target data item already exists in the recipient database, then the (418) `Already exists` exception condition is created by the command.

If there is insufficient space in the recipient database for the data item, then the (420) `Device full` exception condition is created by the command, and the originator SHOULD NOT attempt to add additional data until the recipient has more free space.

Non-specific errors created by the recipient while attempting to complete the command create the (500) `Command failed` exception condition.

If an error occurs while the recipient copying the data item within the recipient's data base, then the (510) `Data store failure` exception condition is created by the command.

Example 1: The client requests the server to copy an item (LUID 1001) from the top level folder to a different folder (LUID 1013), within the same datastore, assigning it the new LUID 1007, within a `Sync` command.

```
<Copy CmdID="301">
  <Item>
    <TargetClientURI>1007</TargetClientURI>
    <SourceClientURI>1001</SourceClientURI>
    <TargetClientParentURI>1013</TargetClientParentURI>
    <SourceClientParentURI>./</SourceClientParentURI>
  </Item>
</Copy>
```

Example 2: The client requests the server to copy an item (LUID 1002) from the top level folder of the ./Contacts datastore to a different folder (LUID 1008), within the ./Archive datastore, assigning it the new LUID 1010, outside of a `Sync` command.

```
<Copy CmdID="302">
```

```
  <Item>
    <TargetClientURI>1010</TargetClientURI>
    <SourceClientURI>1002</SourceClientURI>
    <TargetClientParentURI>./Archive/1008</TargetClientParentURI>
    <SourceClientParentURI>./Contacts</SourceClientParentURI>
  </Item>
</Copy>
```

Example 3: The server requests the client to copy an existing, mapped item (LUID 1001) from the top level folder of one datastore to a different existing, mapped folder (LUID 1008) in the ./Archive datastore, assigning it a GUID of ABC012345_1011, inside of a `Sync` command.

```
<Copy CmdID="303">
  <Item>
    <TargetServerURI>ABC012345_1011</TargetServerURI>
    <SourceClientURI>1001</SourceClientURI>
    <TargetClientParentURI>./Archive/1008</TargetClientParentURI>
    <SourceClientParentURI>./</SourceClientParentURI>
  </Item>
</Copy>
```

Example 4: The server requests the client to copy an existing, mapped item (LUID 1002) from the top level folder of one datastore to a different existing, mapped folder (LUID 1013) in the same datastore, assigning it a GUID of ABC012345_1012, outside of a `Sync` command.

```
<Copy CmdID="304">
  <Item>
    <TargetServerURI>ABC012345_1012</TargetServerURI>
    <SourceClientURI>1002</SourceClientURI>
    <TargetClientParentURI>./Contacts/1004</TargetClientParentURI>
    <SourceClientParentURI>./Contacts</SourceClientParentURI>
  </Item>
</Copy>
```

Example 5: The server requests the client to copy an existing but unmapped item (GUID ABC012345_1003) from the top level folder of the current datastore to a different existing, mapped folder (LUID 1004) in the same datastore, assigning it a GUID of ABC012345_1013, inside of a `Sync` command.

```
<Copy CmdID="305">
  <Item>
    <TargetServerURI>ABC012345_1013</TargetServerURI>
    <SourceServerURI>ABC012345_1003</SourceServerURI>
    <TargetClientParentURI>1004</TargetClientParentURI>
    <SourceClientParentURI>./</SourceClientParentURI>
  </Item>
</Copy>
```

Example 6: The server requests the client to copy an existing but unmapped item (GUID ABC012345_1003) from the top level folder of a datastore to a different existing, mapped folder (LUID 1008) in a different datastore, assigning it a GUID of ABC012345_1014, outside of a `Sync` command.

```
<Copy CmdID="306">
  <Item>
    <TargetServerURI>ABC012345_1014</TargetServerURI>
    <SourceServerURI>ABC012345_1003</SourceServerURI>
```

```
    <TargetClientParentURI>./Archive/1008</TargetClientParentURI>
    <SourceClientParentURI>./Contacts</SourceClientParentURI>
  </Item>
</Copy>
```

Example 7: The server requests the client to copy an existing mapped item (LUID 1001) from the top level folder to an existing, unmapped folder (GUID ABC012345_1005), assigning it a GUID of ABC012345_1015, inside of a `Sync` command.

```
<Copy CmdID="307" >
  <Item>
    <TargetServerURI>ABC012345_1015</TargetServerURI>
    <SourceClientURI>1001</SourceClientURI>
    <TargetServerParentURI>ABC012345_1005</TargetServerParentURI>
    <SourceClientParentURI>./</SourceClientParentURI>
  </Item>
</Copy>
```

Example 8: The server requests the client to copy an existing unmapped item (GUID ABC012345_1003) from the top level folder to an existing, unmapped folder (GUID ABC012345_1005), assigning it a GUID of ABC012345_1016, inside of a `Sync` command.

```
<Copy CmdID="308">
  <Item>
    <TargetServerURI>ABC012345_1016</TargetServerURI>
    <SourceServerURI> GUID ABC012345_1003</SourceServerURI>
    <TargetServerParentURI>ABC012345_1005</TargetServerParentURI>
    <SourceClientParentURI>./</SourceClientParentURI>
  </Item>
</Copy>
```

Example 9: The server requests the client to copy an existing unmapped item (GUID ABC012345_1006) from an existing unmapped folder (GUID ABC012345_1005) to an existing, mapped folder (LUID 1013), assigning it a GUID of ABC012345_1017, inside of a `Sync` command.

```
<Copy CmdID="309">
  <Item>
    <TargetServerURI>ABC012345_1017</TargetServerURI>
    <SourceServerURI>GUID ABC012345_1006</SourceServerURI>
    <TargetClientParentURI>1013</TargetClientParentURI>
    <SourceServerParentURI>GUID ABC012345_1005</SourceServerParentURI>
  </Item>
</Copy>
```

Example 10: The server requests the client to copy an existing unmapped item (GUID ABC012345_1006) from an existing unmapped folder (GUID ABC012345_1005) to an the same existing unmapped folder (GUID ABC012345_1006), assigning it a GUID of ABC012345_1018, inside of a `Sync` command.

```
<Copy CmdID="310">
  <Item>
    <TargetServerURI>ABC012345_1018</TargetServerURI>
    <SourceServerURI>GUID ABC012345_1006</SourceServerURI>
    <TargetServerParentURI>GUID_ABC012345_1005</TargetServerParentURI>
    <SourceServerParentURI>GUID ABC012345_1005</SourceServerParentURI>
  </Item>
```

```
</Copy>
```

6.2.16 Correlator

Usage: Specifies a link between two related `SyncAlert` commands.

Parent Element: `SyncAlert`

Content Model:

```
<xs:attribute name="Correlator" type="xs:string"/>
```

Restrictions:

None

6.2.17 Cred

Usage: Specifies an authentication credential for the originator.

Parent Elements: `Add`, `Alert`, `Copy`, `Delete`, `Get`, `Put`, `Move`, `Replace`, `Status`, `Sync`, `SyncAlert`, `SyncHdr`

Content Model:

Content Model:

```
<xs:element name="Cred" type="CredType"/>
<xs:complexType name="CredType">
  <xs:sequence>
    <xs:element ref="Meta"/>
    <xs:element ref="Data"/>
  </xs:sequence>
  <xs:attribute ref="AuthName" use="optional">
</xs:complexType>
```

Restrictions:

The `Meta` *`Type`* and *`Format`* attributes specify the authentication scheme and credential encoding style, respectively. The value of the `Data` element specifies the credential. For authentication schemes which do not contain an extractable user identifier, the *`AuthName`* attribute is used to hold a user specific identifier.

Credentials SHOULD be processed from the `SyncML` message level (supplied on the `SyncHdr`), down to the datastore level (supplied on a `SyncAlert` or `Sync` command), down to the individual command. Credential failure at the message level should stop lower level credential checking until that has been resolved.

The Data Sync Client and Server MUST support the following authentication scheme:

Name	Authentication Scheme	Description
`SHA-256`	`syncml:auth-sha256`	`SHA-256 hash-function based authentication scheme.`

SHA-256 authentication scheme provides a safe way for prevention of replay attacks to transmit the credential in SHA-256 digest to the recipient. When SHA-256 authentication scheme is used, the value of `Meta` *`Type`* and *`Format`* attributes SHALL be 'syncml:auth-sha256' and 'b64', respectively. The value of `Data` element is the digest which SHALL be computed as following:

Let H = the SHA-256 Hashing function.

Let Digest = the output of the SHA-256 Hashing function.

Let B64 = the base64 encoding function.

Let userid = User Identifier.

Let secret = Secrete known by the originator and recipient.

Let nonce = Challenge specified by the recipient

Digest = H(B64(H(userid:secret)):nonce)

If absent, and no other authentication credential was specified in either a parent command or in the `SyncHdr` element, then no authentication credential is specified.

If an authentication credential was specified by a parent command or in the `SyncHdr` element, then that authentication credential specified there is assumed to be sufficient for the operation specified by the current element. Specifying insufficient authentication credentials will result in a '(`401`) `Unauthorized`' exception condition.

If the authentication challenge is received for the request, the authentication scheme and encoding of the next request SHALL be applied to it.

The DS Client and Server MAY support the following authentication schemes (not the definitive list):

Name	Authentication Scheme	Description
SHA-1	syncml:auth-sha1	SHA-1 hash-function based authentication scheme.
MD5	syncml:auth-md5	MD5 hash-function based authentication scheme.
X509	syncml:auth-x509	The data would be an actual X.509 Certificate. The data SHOULD be sent raw in WBXML, and base64 encoded in XML.
Securid	syncml:auth-securid	The data specific for SecurID authentication would be sent. The data SHOULD be sent raw in WBXML, and base64 encoded in XML.
Safeword	syncml:auth-safeword	The data specific for SafeWord authentication would be sent. The data SHOULD be sent raw in WBXML, and base64 encoded in XML.
Digipass	syncml:auth-digipass	The data specific for DigiPass authentication would be sent. The data SHOULD be sent raw in WBXML, and base64 encoded in XML.

Other authentication schemes MAY be specified by prior agreement between the originator and the recipient, which is out of the scope of this document.

Example: The following is an example of a SHA-256 authentication scheme:

```
<Cred AuthName="Bruce">
  <Meta Format="b64" Type="syncml:auth-sha256"/>
  <Data>  --actual data goes here-- </Data>
</Cred>
```

6.2.18 Data

Usage: Specifies discrete SyncML data.

Parent Elements: Cred, Item

Content Model:

```
<xs:element name="Data" type="DataType"/>
<xs:complexType name="DataType">
  <xs:simpleContent>
    <xs:extension base="xs:string">
      <xs:attribute ref="Encrypted" use="optional"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
```

Restrictions:

The content information can be either parsable character data or mark-up data. If the element type contains any mark-up, then the name space for the element types MUST be declared on the element types in the content information.

The optional *Encrypted* attribute specifies whether or not the content data is encrypted.

When specified in a Cred, the element type specifies the authentication credentials.

When specified in an Item, the element type specifies the item data.

Example 1: The following is an example of an Item with data that does not contain any mark-up.

```
<Item>
  <Data>John Smith, +1-919-555-1234</Data>
</Item>
```

Example 1: The following is an example of an Item with data that does contain meta-information mark-up data.

```
<Item>
  <Meta Format="xml" Type="application/vnd.syncml-devinf+xml"/>
  <Data><![CDATA[
    <DevInf xmlns='syncml:devinf' Version="2.0">
      <DevCap>
        <Man>HAL</Man>
        <Model>9000</Model>
        <OEM>Classified</OEM>
        <FwV>1.0Beta</FwV>
        <SwV>1.0Beta</SwV>
        <HwV>1.0Beta</HwV>
        <DevID>HAL 9000</DevID>
        <DevType>server</DevType>
      </DevCap>
      <DataStore>
        ..
      </DataStore>
    </DevInf>]]>
  </Data>
</Item>
```

6.2.19 Delete

Usage: Specifies the SyncML command to delete data from a data collection.

Parent Elements: Sync, SyncBody

Content Model:

```
<xs:element name="Delete" type="DeleteType"/>
<xs:complexType name="DeleteType">
  <xs:sequence>
    <xs:element ref="Cred" minOccurs="0"/>
    <xs:element ref="Item" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute ref="CmdID" use="required"/>
  <xs:attribute ref="SftDel" use="optional"/>
  <xs:attribute ref="NoStatus" use="optional"/>
</xs:complexType>
```

Restrictions:

The Delete command is generally used to permanently erase data items from the recipient's database. However, the command can also be used to temporarily remove data items from the recipient's database in order to create room for a subsequent Add command. This is termed a "soft delete".

One or more Item element types MUST be specified. The Item element type specifies the data item deleted from the database. The TargetClientURI or SourceClientURI specified within the Item element type is a client identifier (LUID), as relative to the corresponding TargetServerURI / TargetClientURI or SourceServerURI / SourceClientURI specified in the parent Sync command.

TargetServerURI, SourceServerURI, TargetServerParentURI, TargetClientParentURI, SourceServerParentURI or SourceClientParentURI MUST NOT be specified in a Delete command. If the client wishes to delete an item that has not yet been mapped, it must first send the new LUID in the corresponding StatusItem element of the item to be deleted. The Server MUST NOT attempt to delete an item before the client has had an opportunity to send a new LUID in the StatusItem element (E.g. Adding an item, and then immediately deleting the same item by the GUID is prohibited).

In applications (e.g., email) where the "delete" concept involves "moving" a data item from one folder to a special "Deleted" folder, this can be achieved either by the Move command, or by using either the Copy command or the Add command to propagate the specified data item to the "Deleted" folder, followed by the subsequent Delete of the corresponding item from the original folder.

The recipient of a Delete command can delete any subset of the specified data elements. However, if all of the requested data was not deleted, then the (206) Partial content exception condition is created by the command. If a Status command is returned for this exception condition, then the identifiers of the data items not deleted SHOULD be returned also.

If the recipient determines that the data item doesn't exists on the recipient's database, then the (211) Item not deleted exception condition is created by the command.

If the originator's authentication credentials specify a principal with insufficient rights to complete the command, then the (401) Unauthorized exception condition is created by the command. If no authentication credentials were specified, then (407) Authentication required exception condition is created by the command. A suitable challenge can also be returned.

In synchronization protocol cases where the client sends a new LUID for the GUID in a StatusItem element, the server MUST always specify the client identifier for any data items to be deleted. Otherwise, the (412) Incomplete command exception condition is created and no data items will be deleted by the client.

The Delete command can be used to delete an interior node and all of its child nodes. If a Delete is received for a parent node which still contains child items, the child items are also deleted.

Non-specific errors created by the recipient while attempting to complete the command create the (500) Command failed exception condition.

Example 1: The client requests the server to delete an item, with credentials that provide delete access to the item in the current datastore. The item is identified by the client LUID.

```
<Delete CmdID="401">
  <Cred>
    <Meta Type="syncml:auth-md5" Format="b64"/>
    <Data>Zz6EivR3yeaaENcRN6lpAQ==</Data>
  </Cred>
  <Item><SourceClientURI>1001</SourceClientURI></Item>
</Delete>
```

The following is an example to "soft delete" a number of data items to allow room on the device for a subsequent `Add` or `Copy` command, not specified in this example.

Example 2: The client informs the server that it has Soft Deleted a set of items to free storage on the device. The items are identified by client LUIDs.

```
<Delete CmdID="402" SftDel="true">
  <Item><SourceClientURI>1015</SourceClientURI></Item>
  <Item><SourceClientURI>1016</SourceClientURI></Item>
  <Item><SourceClientURI>1017</SourceClientURI></Item>
</Delete>
```

Example 3: The server requests the client to Soft Delete a set of items to free storage on the device. The items are identified by client LUIDs.

```
<Delete CmdID="403" SftDel="true">
  <Item><TargetClientURI>1018</TargetClientURI></Item>
  <Item><TargetClientURI>1019</TargetClientURI></Item>
  <Item><TargetClientURI>1020</TargetClientURI></Item>
</Delete>
```

6.2.20 Direction

Usage: Specifies the sync direction.

Parent Element: `SyncType`

Content Model:

```
<xs:attribute name="Direction">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="fromClient"/>
      <xs:enumeration value="fromServer"/>
      <xs:enumeration value="twoWay"/>
      <xs:enumeration value="NoWay"/>
    </xs:restriction>
  </xs:simpleType>
</xs:attribute>
```

Restrictions:

The *`Direction`* attribute specify the sync direction.

The semantics of *`Direction`*, *`Behaviour`*, *`IDValidity`* and *`ChangeLogValidity`* are described in [DSPRO].

6.2.21 EmptyType

Usage: This type definition is used in the Syntax Elements definition. And it is used to specify an empty type for the Element that has no element, attribute or content.

Used for Elements: `Final`, `MoreData`

Content Model:

```
<xs:complexType name="EmptyType"/>
```

Example:

```
<xs:element name="Final" type="EmptyType" />
```

6.2.22 Encrypted

Usage: Specifies whether or not the data is encrypted.

Parent Element: `Data`

Content Model:

```
<xs:attribute name="Encrypted" type="xs:boolean" default="false"/>
```

Restrictions:

This attribute specifies whether or not the content information in the `Data` element is encrypted using the encrypted symmetry key transmitted in `SyncHdr`.

Example: The following is an example of an Item with data that does not contain any mark-up.

```
<Item>
  ...  ...
  <Data Encrypted="true">blahblahblah</Data>
</Item>
```

6.2.23 EncryptedKey

Usage: Specifies encrypted symmetric key and key information for the originator.

Parent Element: `SyncHdr`

Content Model:

```
<xs:element name="EncryptedKey" type="CredType"/>
```

Restrictions:

The `Meta` *`Type`* and *`Size`* attributes specify the algorithm and length of the symmetric key, respectively. The `Meta` *`Format`* attribute specifies the encoding style for the encrypted octet sequence. The default value for this element is b64. The value of the `Data` element specifies the encrypted symmetric key using the key exchange algorithm.

Example: The following example illustrates how the encrypted symmetric key is transmitted.

```
<SyncHdr>
  ...
  <EncryptedKey>
    <Meta Format="b64" Type="AES-128-CBC" Size="128"/>
    <Data>Zz6EivR3yeaaENcRN6lpAQ==</Data>
```

```
  </EncryptedKey>
</SyncHdr>
```

6.2.24 Field

Usage: Specifies a field level filter to be performed on the parent element of the `Filter` element.

Parent Element: `Filter`

Content Model:

```
<xs:element name="Field" type="FieldType"/>
<xs:complexType name="FieldType">
  <xs:sequence>
    <xs:element ref="Item"/>
  </xs:sequence>
</xs:complexType>
```

Restrictions:

If the `Field` element is present, the `SyncAlert TargetClientURI/TargetServerURI Meta` Type is used to indicate the content type used in the content filtering and MUST be present. The `Item Meta` element is used to indicate the device info MIME type and MUST be present. The `Item Data` element MUST contain `Property` elements. The mark-up characters of the `Data` element content MUST be properly escaped according to [XML] specification rules or the CDATA sections MUST be used. The `Property` elements MUST be used to override any `Property` elements previously received in the `CTCap` element for the content-type being filtered and MUST apply to the current synchronization session only. If no `Field` element is present in the `Filter` element, then all properties SHOULD be filtered using the device info data store `CTCap` element for the specified `SyncAlert` element.

Example: The following is an example of a `Field` element used within a `Filter` element to define the characteristics of the subset of data to be synchronized. The `Field` element contains a Property element set to "PHOTO" containing a MaxSize element set to 0 (zero). This indicates to the server that it SHOULD NOT send any PHOTO properties since the client has requested that it wishes to receive only 0 bytes of this property for this synchronization request and the value SHOULD NOT be truncated.

```
<Filter>
  ...
  <Field>
    <Item>
      <Meta Type="application/vnd.syncml-devinf+xml"/>
      <Data><![CDATA[
        <Property>
          <PropName>PHOTO</PropName>
          <PropInfo>
            <MaxSize Truncate="false">0</MaxSize>
          </PropInfo>
        </Property>
      ]]></Data>
    </Item>
  </Field>
  ...
</Filter>
```

6.2.25 FieldLevel

Usage: Indicates that the content information in the `Data` element replaces only part of an item.

Parent Element: `Replace`

Content Model:

```
<xs:attribute name="FieldLevel" type="xs:boolean" default="false"/>
```

Restrictions:

Example:

```
<Replace CmdID="3" FieldLevel="true">
  <Meta Type="x-type/x-subtype"/>
  <Item>
    <TargetClientURI>244</TargetClientURI>
    <Data>
      ...
    </Data>
  </Item>
</Replace>
```

6.2.26 Filter

Usage: Specifies a filter action to be performed on the parent element.

Parent Element: `Get`, `SyncAlert`

Content Model:

```
<xs:element name="Filter" type="FilterElementType"/>
<xs:complexType name="FilterElementType">
  <xs:sequence>
    <xs:element ref="Meta"/>
    <xs:element ref="Field" minOccurs="0"/>
    <xs:element ref="Record" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute ref="FilterType" use="optional"/>
</xs:complexType>
```

Restrictions:

The `Filter` element MAY appear in the `SyncAlert` element. If the `Filter` element is present, the `Meta` element's *`Type`* attribute is used to indicate the content type used in the filter query and MUST be present. If the `Filter` element does not have a `Record` or a `Field` element, then the filter request is ignored and synchronization MUST continue without any filtering.

Example: The following is an example of a `Filter` element which

1. Uses the `Record` element with a `Meta` element's *`Type`* value to specify the MIME type the filter applies to.
2. Uses the `Item` element with a `Meta` element's *`Type`* value of "syncml:filtertype-cgi" to indicate the grammar being used.
3. Uses the `Item Data` element to constrain the items synchronized to those that fall into the "business" or "personal" group (case insensitive) with the cgi expression "GROUP&iCON;business&OR; GROUP &iCON;personal".

```
<Filter>
  <Meta Type="Some/MIMEType"/>
  <Record>
    <Item>
      <Meta Type="syncml:filtertype-cgi"/>
      <Data><![CDATA[GROUP&iCON;business&OR;GROUP&iCON;personal]]></Data>
    </Item>
  </Record>
</Filter>
```

6.2.27 FilterType

Usage: Indicates the type of filtering behaviour that is being requested. If the requested filter type is not supported by the recipient then a Status code 406 (OPTIONAL feature not supported) MUST be returned. The `Item` element of the `Status` command SHOULD indicate that the *`FilterType`* attribute was the unsupported feature.

Parent Element: `Filter`

Content Model:

```
<xs:attribute name="FilterType" default="EXCLUSIVE"/>
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="EXCLUSIVE"/>
      <xs:enumeration value="INCLUSIVE"/>
    </xs:restriction>
  </xs:simpleType>
</xs:attribute>
```

Restrictions:

If present, these keywords MUST be one of the *`FilterType`* keywords listed below. If not present, then the *`FilterType`* value of "EXCLUSIVE" MUST be assumed.

Keywords	Description
EXCLUSIVE	Indicates that the sender is requesting that the set of data items to be synchronized MUST be exactly the set of items specified by the Filter. Additional items in the data store MUST not be synchronized and the recipient of the Filter MUST request that additional items be removed from the sender device.
INCLUSIVE	Indicates that the sender is requesting that the set of data items to be synchronized MUST include the set of items specified by the Filter. Additional items in the datastore that are not specified by the Filter MUST not be synchronized and the recipient of the Filter MUST leave these additional items on the sender device.

Table 1: FilterType keywords

Example: The following is an example of a `Filter` element which uses the EXCLUSIVE *`FilterType`* keyword

```
<Filter FilterType="EXCLUSIVE">
  <Meta Type="Some/MIMEType"/>
  <Record>
    <Item>
      <Meta Type="syncml:filtertype-cgi"/>
      <Data><![CDATA[GROUP&iCON;business&OR;GROUP&iCON;personal]]></Data>
    </Item>
  </Record>
</Filter>
```

6.2.28 Final

Usage: Indicator that the `SyncML` message is the last message in the current SyncML package.

Parent Element: SyncBody

Content Model:

```
<xs:element name="Final" type="EmptyType"/>
```

Restrictions:

The element type MUST only be specified on the last message of the SyncML package. If not present, then more messages follow this SyncML message in the current SyncML package.

The OMA DS Protocol specification [DSPRO] specifies the semantics of the different SyncML packages.

Example:

```
<SyncML ... >
  <SyncHdr>...blah, blah...</SyncHdr>
  <SyncBody>
    ...blah, blah...
    <Final/>
  </SyncBody>
</SyncML>
```

6.2.29 Format

Usage: Specifies the encoding format of the content information in the Data element.

Parent Element: Meta

Content Model:

```
<xs:attribute name="Format" type="xs:string"/>
```

Restrictions:

The value of this attribute SHOULD be one of bin, bool, b64, chr, int, node, null, xml, date, time, or float. If this attribute type is missing, the default value is chr. If the value is bin, then the format of the content is binary data. If the value is bool, then the format of the content is either true or false. If the value is b64, then the format of the content information is binary data that has been character encoded using the Base64 transfer encoding defined by [RFC2045]. If the value is chr, then the format of the content information is clear-text in the character set specified on the transport protocol, the MIME content type header or the XML prolog. If the value is int, then the format of the content information is numeric text representing the integer. If the value is null, then there is no content information. This value is used by some synchronization data models to delete the content, but not the presence of the property. If the value is xml, then the format of the content information is XML structured mark-up data. If the value is date, then the format of the content is in ISO 8601 format with the century being included in the year [ISO8601]. If the value is time, then the format of the content is in ISO 8601 format. If the value is float, then the format of the content is standard concept of real numbers corresponding to a single precision 32 bit floating point type as defined in XML Schema 1.0 as the float primitive type.

In case a Meta element containing a *Format* attribute contains meta-information about a Data object, this Meta element MUST have the same parent as the Data object it refers to.

The target object is the one in which the meta-information appears.

Example: The following example illustrates how the attribute is used to specify *Format* meta-information for data in the Item element type.

```
<Item>
  <Meta Format="int" Type="text/plain" />
  <Data>1024</Data>
</Item>
```

6.2.30 FP

Usage: Fingerprints are values associated with particular data item contents. And fingerprints are compared to detect if particular data items have changed.

Parent Elements: `SourceClientURI`, `ID`

Content Model:

```
<xs:attribute name="FP" type="xs:int "/>
```

Restrictions:

None.

Example:

```
<Replace CmdID="3">
  ... ...
  <Item>
    <SourceClientURI FP="0568">244</SourceClientURI>
    <Data>
      ...
    </Data>
  </Item>
</Replace>
```

6.2.31 FreeID

Usage: Specifies the number of free item identifiers available for adding new items to the datastore.

Parent Element: `Sync`

Content Model:

```
<xs:attribute name="FreeID" type="xs:unsignedInt"/>
```

Restrictions:

The content information is a decimal integer number.

Example: The following is an example showing 25 free item identifiers

```
FreeID="25"
```

6.2.32 FreeMem

Usage: Specifies the amount of free memory, in bytes, available in the datastore.

Parent Element: `Sync`

Content Model:

```
<xs:attribute name="FreeMem" type="xs:long"/>
```

Restrictions:

The content information is a decimal integer number.

Example: The following is an example showing 1022 free bytes

```
FreeMem="1022"
```

6.2.33 Get

Usage: Specifies the SyncML command to retrieve data from the recipient.

Parent Element: SyncBody

Content Model:

```
<xs:element name="Get" type="GetType"/>
<xs:complexType name="GetType">
  <xs:sequence>
    <xs:element ref="Cred" minOccurs="0"/>
    <xs:element ref="Meta" minOccurs="0"/>
    <xs:element ref="Item" maxOccurs="unbounded"/>
    <xs:element ref="Filter" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute ref="CmdID" use="required"/>
  <xs:attribute ref="NoStatus" use="optional"/>
</xs:complexType>
```

Restrictions:

There is no synchronization state information for data retrieved using the Get command. This command MUST NOT be specified within a Sync command.

Data returned from a Get command is returned in a Results element type in a subsequent SyncML message.

One or more Item element types MUST be specified. The Item element type specifies the data items to be returned from the recipient. The TargetServerURI / TargetClientURI and SourceServerURI / SourceClientURI specified within the Item element type SHOULD be an absolute URI.

The optional Filter element specifies the filter criteria applied to the TargetServerURI/TargetClientURI in each Item element.

If the command completed successfully, then the (200) OK exception condition is created by the command.

If the command completed successfully but there is no content to return, then the (204) No content exception condition is created by the command.

If the command completed successfully but only a portion of the content is being returned, with the remainder being returned in subsequent Results commands, then the (206) Partial content exception condition is created by the command.

If the command specifies an ambiguous target with multiple matches, then the (300) Multiple choices exception condition is created by the command.

If the originator's authentication credentials specify a principal with insufficient rights to complete the command, then the (401) Unauthorized exception condition is created by the command. If no authentication credentials were specified, then (407) Authentication required exception condition is created by the command. A suitable challenge can also be returned.

If the specified data item doesn't exist on the recipient, then the (404) Not found exception condition is created by the command.

If the requested data item is too large to be transferred at this time, then the (413) Request entity too large exception condition is created by the command.

Non-specific errors created by the recipient while attempting to complete the command create the (500) Command failed exception condition.

If the MIME content type or content format for the data item is not supported by the recipient, then the (415) Unsupported MIME content type or content format exception condition is created by the command.

Example:

```
<Get CmdID="12345">
  <Cred>
    <Meta Type="syncml:auth-md5" Format="b64"/>
    <Data>Zz6EivR3yeaaENcRN6lpAQ==</Data>
  </Cred>
  <Item>
    <TargetClientURI>./telecom/pb</TargetClientURI>
    <SourceServerURI>http://www.datasync.com/servlet/</SourceServerURI>
    <Meta Type="text/x-vCard"/>
  </Item>
</Get>
```

6.2.34 ID

Usage: Specifies the identifier of the data item, and optionally specifies the fingerprint of the data item.

Parent Element: IDContainer

Content Model:

```
<xs:element name="ID" type="IDType"/>
<xs:complexType name="IDType">
  <xs:simpleContent>
    <xs:extension base="xs:string">
      <xs:attribute ref="FP" use="optional"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
```

Restrictions:

The ID element specifies the identifier of the data item, and the optional *FP* attribute specifies the fingerprint of the data item.

Example:

```
<ID FP="01">LUID001</ID>
```

6.2.35 IDContainer

Usage: Acts as the placeholder element for one or more ID elements.

Parent Element: SyncAlert

Content Model:

```
<xs:element name="IDContainer" type="IDContainerType"/>
<xs:complexType name="IDContainerType">
  <xs:sequence>
    <xs:element ref="ID" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

Restrictions:

The `IDContainer` element is used to specify one or more data item identifiers. By using the `IDContainer` element in a `SyncAlert` element, the sender can indicate the data item identifiers that the sender wants to send to the receiver, and the receiver can return back the data item identifiers that the receiver wants the sender to send.

Example:

```
<SyncAlert CmdID="3">
  ... ...
  <IDContainer>
    <ID FP="01">LUID001</ID>
    <ID FP="02">LUID002</ID>
    <ID FP="03">LUID003</ID>
  </IDContainer>
  ...
</SyncAlert>
```

6.2.36 IDValidity

Usage: Specifies if the data item IDs are valid.

Parent Element: `SyncType`

Content Model:

```
<xs:attribute name="IDValidity" type="xs:boolean"/>
```

Restrictions:

The *`IDValidity`* attribute specifies if the ID is valid.

The semantics of *`Direction`*, *`Behaviour`*, *`IDValidity`* and *`ChangeLogValidity`* are described in [DSPRO].

6.2.37 Item

Usage: Specifies a container for item data.

Parent Elements: `Add`, `Alert`, `Copy`, `Delete`, `Field`, `Get`, `Put`, `Move`, `Record`, `Replace`, `Results`, `Status`

Content Model:

```
<xs:element name="Item" type="ItemType"/>
<xs:complexType name="ItemType">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="TargetClientURI" minOccurs="0"/>
      <xs:element ref="TargetServerURI" minOccurs="0"/>
    </xs:choice>
    <xs:choice>
      <xs:element ref="SourceClientURI" minOccurs="0"/>
      <xs:element ref="SourceServerURI" minOccurs="0"/>
    </xs:choice>
    <xs:choice>
      <xs:element ref="TargetClientParentURI" minOccurs="0"/>
      <xs:element ref="TargetServerParentURI" minOccurs="0"/>
    </xs:choice>
    <xs:choice>
      <xs:element ref="SourceClientParentURI" minOccurs="0"/>
      <xs:element ref="SourceServerParentURI" minOccurs="0"/>
```

```
    </xs:choice>
    <xs:element ref="Meta" minOccurs="0"/>
    <xs:element ref="Data" minOccurs="0"/>
    <xs:element ref="MoreData" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

Restrictions:

If the source URI for the data is an external entity, then the `Data` element is absent. In this case, the recipient will need to retrieve the data from the specified network location.

The `TargetClientURI/TargetServerURI` or `SourceClientURI/SourceServerURI` element types for any of the `SyncML` commands can be a relative URL. This restriction is not captured by the `SyncML` Schema.

When specified in an `Add`, `Copy`, `Delete`, `Get`, `Put`, `Replace`, or `Results` command, the element type specifies the data item that is the operand for the command. One or more `Item` element types MUST be specified in these commands. The `Item` element type specifies the data item to be operated on the recipient's database. When these commands are specified within a `Sync` element type, the `TargetServerURI/SourceServerURI` within the `Item` element type SHOULD be a global unique identifier (GUID), and the `TargetClientURI/SourceClientURI` within the `Item` element type SHOULD be a local unique identifier (LUID), as relative to the corresponding `TargetServerURI/SourceServerURI` and `TargetClientURI/SourceClientURI` specified in the parent `Sync` command. If specified within a `SyncBody` element type, the `TargetClientURI/TargetServerURI` and `SourceClientURI/SourceServerURI` within the `Item` element type in these commands SHOULD be an absolute URI.

When specified in an `Alert`, the element type specifies the parameters for the alert type.

When specified in a `Status`, the element type specifies additional information about the request status code type. For example, it might specify the component of the request that caused the status condition.

Example:

```
<Add CmdID="1" >
  <Cred>
    <Meta Type="syncml:auth-md5" Format="b64"/>
    <Data>Zz6EivR3yeaaENcRN6lpAQ==</Data>
  </Cred>
  <Item>
    <SourceClientURI>15</SourceClientURI>
    <Meta Type="text/directory profile=vCard"/>
    <Data>BEGIN:VCARD
VERSION:3.0
FN:Smith;Bruce
N:Bruce Smith
TEL;TYPE=WORK;VOICE:+1-919-555-1234
END:VCARD
    </Data>
  </Item>
</Add>
```

6.2.38 Last

Usage: Specifies the synchronization state information (i.e., sync anchor) for the previous synchronization session.

Parent Element: `Anchor`

Content Model:

```
<xs:attribute name="Last" type="AnchorType"/>
<xs:simpleType name="AnchorType">
```

```
  <xs:union memberTypes="xs:dateTime xs:positiveInteger"/>
</xs:simpleType>
```

Restrictions:

The OPTIONAL *Last* attribute specifies the synchronization anchor for the previous synchronization session.

The value of the *Last* attribute MUST specify either an UTC based date/time stamp or a monotonically increasing numeric integer. If a date/time stamp, then the text MUST be in the complete representation, basic format defined by ISO8601.

All *Last* and *Next* values sent in a synchronization session by a particular sender MUST be of the same type, and MUST be used such that a comparison operation on values can determine older from newer.

Determination of the ordinal sequence of the version of an existing object in the recipient and the version of the object can be made by comparing the content information of the object with the value on the existing object.

Example:

```
<Anchor Last="2000-08-24T13:30:00Z" Next="2000-08-24T22:13:00Z"/>
```

6.2.39 MaxMsgSize

Usage: Specifies the maximum byte size of any response message to a given SyncML request.

Parent Element: SyncHdr

Content Model:

```
<xs:attribute name="MaxMsgSize" type="xs:long"/>
```

Restrictions:

The attribute appears in the SyncHdr of a SyncML request to specify the maximum size of any subsequent response messages. The attribute is usually specified by a SyncML client, but can also be specified by a SyncML server.

This attribute value is applicable for the remainder of the synchronization session, unless it is specified again.

The attribute value represents the maximum, decimal byte size of any response message.

Example:

```
MaxMsgSize="1023"
```

6.2.40 MaxObjSize

Usage: Specifies the maximum size in bytes of a data object that the device is able to receive.

Parent Elements: SyncAlert, Sync

Content Model:

```
<xs:attribute name="MaxObjSize" type="xs:unsignedLong"/>
```

Restrictions:

The attribute appears in a SyncML request to specify the maximum size of the largest object it is capable of receiving in any subsequent response messages. This attribute value is applicable for the remainder of the synchronization session.

The attribute value represents the maximum, decimal byte size without leading zeroes of any object.

Example: Device that can receive a maximum object of 10K bytes.

```
MaxObjSize="10240"
```

6.2.41 Meta

Usage: Specifies meta-information about the parent element type.

Parent Elements: `Add`, `Chal`, `Copy`, `Cred`, `Delete`, `Get`, `Filter`, `Item`, `Move`, `Put`, `Replace`, `Results`, `Sync`

Content Model:

```
<xs:element name="Meta" type="MetaType"/>
<xs:complexType name="MetaType">
  <xs:sequence>
    <xs:attribute ref="Format" use="optional"/>
    <xs:attribute ref="Type" use="required"/>
    <xs:attribute ref="Size" use="optional"/>
  </xs:sequence>
</xs:complexType>
```

Restrictions:

When specified in the `Chal` element, the `Meta` specifies meta-information about the authentication scheme requested.

When specified in the `Cred` element, the `Meta` specifies meta-information about the authentication credential.

When specified in the `Sync` element, the scope for the meta-information includes all the contained commands, unless the meta-information is overridden by a `Meta` in a contained command.

When specified in the `Results` element, the `Meta` specifies meta-information about the results set.

When specified in the `Add`, `Copy`, `Delete`, `Get`, and `Replace` commands, the element type specifies meta-information about the `SyncML` command. For example, the common MIME content type or content format for all the specified items. The scope of the `Meta` information is limited to the command.

Example:

```
<Item>
  <Meta Format="xml" Type="application/vnd.syncml-devinf+xml"/>
  <Data><![CDATA[
    <DevInf xmlns='syncml:devinf' Version="2.0">
      <DevCap>
        <Man>HAL</Man>
        <Model>9000</Model>
        <OEM>Classified</OEM>
        <FwV>1.0Beta</FwV>
        <SwV>1.0Beta</SwV>
        <HwV>1.0Beta</HwV>
        <DevID>HAL 9000</DevID>
        <DevType>server</DevType>
      </DevCap>
      <DataStore>
        ..
      </DataStore>
    </DevInf>]]>
  </Data>
</Item>
```

6.2.42 MoreData

Usage: Indicator that a SyncML Data element is incomplete and there will be one or more subsequent chunks.

Parent Element: Item

Content Model:

```
<xs:element name="MoreData" type="EmptyType"/>
```

Restrictions:

The element type MUST be specified on all but the last chunk of Data of an item. If not present, then the item is either contained within a single message or is the closing chunk of the Data item.

Example:

```
<Add CmdID="15" >
  <Meta Type="text/x-vcard" Size="3000"/>
  <Item>
    <SourceClientURI>2</SourceClientURI>
    <Data>BEGIN:VCARD
VERSION:2.1
FN:Bruce Smith
N:Smith;Bruce
TEL;WORK;VOICE:+1-919-555-1234
TEL;WORK;FAX:+1-919-555-9876
NOTE: here starts a huge note field, or icon etc...
    </Data>
    <MoreData/>
  </Item>
</Add>
```

6.2.43 Move

Usage: Specifies a SyncML command to support moving data items from one location (data store and/or folder) to another in the recipient's database.

Parent Elements: Sync, SyncBody

Content Model:

```
<xs:element name="Move" type="CommonOperationType"/>
```

Restrictions:

The Move command allows for moving items (ex: files, folder, emails, vCards) from their current location to a new location.

The behaviour when the source parent folder and the target parent folder are the same is undefined.

The restrictions, valid source/target combinations, possible errors, and example syntax are identical to the Copy command, with the exception that the original item is no longer present after the Move has occurred, and certain modified status codes shown below.

See Copy.

Status Code differences from Copy:

If the command completed successfully, then the (200) OK exception condition is created by the command, instead of (201) Item added.

If the recipient determines that the data item could not be moved on the recipient's database, then the (428) Move Failed exception condition is created by the command.

Example: The following is an example of a Move command similar to example 10 from Copy, except with a different set of directories, so as to avoid attempting to move an item to its current position. The server requests the client to move an existing unmapped item (GUID ABC012345_1006) from an existing unmapped folder (GUID ABC012345_1005) to a different existing unmapped folder (GUID ABC012345_TBD, not shown in Appendix B), assigning it a GUID of ABC012345_1019, inside of a Sync command.

```
<Move CmdID="410">
  <Cred>
    <Meta Type="syncml:auth-md5" Format="b64"/>
    <Data>Zz6EivR3yeaaENcRN6lpAQ==</Data>
  </Cred>
  <Meta Format="chr" Type="text/x-vcard"/>
  <Item>
    <TargetServerURI>ABC012345_1018</TargetServerURI>
    <SourceServerURI>GUID ABC012345_1006</SourceServerURI>
    <TargetServerParentURI>GUID_ABC012345_TBD</TargetServerParentURI>
    <SourceServerParentURI>GUID ABC012345_1005</SourceServerParentURI>
  </Item>
</Move>
```

6.2.44 MsgID

Usage: Specifies a SyncML session-unique identifier for the SyncML Message.

Parent Element: SyncHdr

Content Model:

```
<xs:attribute name="MsgID" type="xs:positiveInteger"/>
```

Restrictions:

The message identifier MUST be unique to the device within the SyncML session. The attribute MUST be specified in the SyncHdr. The value is a monotonically increasing numeric value starting at one (1) for the first message from each device (Client or Server) in the SyncML session. The message identifier specified in a SyncML request MUST be the content of the *MsgRef* element type in the corresponding SyncML Results or response Status.

Example:

```
<SyncHdr SessionID="1" MsgID="1">
  <TargetServerURI>http://syncml.example.com/</TargetServerURI>
  <SourceClientURI>IMEI:001004FF1234567</SourceClientURI>
</SyncHdr>
```

6.2.45 MsgRef

Usage: Specifies a reference to a SyncML session-unique identifier referenced by a SyncML Results or response Status.

Parent Elements: Results, Status

Content Model:

```
<xs:attribute name="MsgRef" type="xs:positiveInteger"/>
```

Restrictions:

The value MUST reference the message identifier of the SyncML message referred to by the Results or response Status.

Example:

```
<Status CmdID="4321" MsgRef="1" CmdRef="1234" Cmd="Add" Code="200"/>
```

6.2.46 Next

Usage: Specifies the synchronization state information (i.e., sync anchor) for the current synchronization session.

Parent Element: Anchor

Content Model:

```
<xs:attribute name="Next" type="AnchorType"/>
<xs:simpleType name="AnchorType">
  <xs:union memberTypes="xs:dateTime xs:positiveInteger"/>
</xs:simpleType>
```

Restrictions:

The REQUIRED *Next* attribute specifies the synchronization anchor for the current synchronization session.

The value of the *Next* attribute MUST specify either an UTC based date/time stamp or a monotonically increasing numeric integer. If a date/time stamp, then the text MUST be in the complete representation, basic format defined by ISO8601.

All *Last* and *Next* values sent in a synchronization session by a particular sender MUST be of the same type, and MUST be used such that a comparison operation on values can determine older from newer.

Determination of the ordinal sequence of the version of an existing object in the recipient and the version of the object can be made by comparing the content information of the object with the value on the existing object.

Example:

```
<Anchor Last="2000-08-24T13:30:00Z" Next="2000-08-24T22:13:00Z"/>
```

6.2.47 NextNonce

Usage: Specifies the nonce string to be used in any subsequent communication.

Parent Element: Chal

Restrictions:

The nonce string MUST be further re-formatted using the Base64 algorithm. Terminators or length of Nonce String MUST NOT be included in this re-formatting. The Nonce string MUST be treated as opaque data.

This element type is used to specify the next nonce string that is to be used in any subsequent SyncML message. For example, a SyncML server specifies this element type to tell the SyncML client to change its nonce to a new value.

Nonce strings are used in various authentication schemes, such as “syncml:auth-sha256” and “syncml:auth-md5”.

Content Model:

```
<xs:element name="NextNonce" type="xs:string"/>
```

Example:

```
<Chal>
```

```
  <Meta Format="b64" Type="syncml:auth-sha256"/>
  <NextNonce>Tm9uY2U=</NextNonce>
</Chal>
```

6.2.48 NoStatus

Usage: Indicates that the originator does not want a response `Status` sent back in the response message.

Parent Elements: `Add`, `Alert`, `Copy`, `Delete`, `Get`, `Move`, `Put`, `Replace`, `Sync`, `SyncAlert`, `SyncHdr`

Content Model:

```
<xs:attribute name="NoStatus" type="xs:boolean" default="false"/>
```

Restrictions:

When specified as "true", the recipient MUST NOT return a `Status` command for the associated `SyncML` command. If specified on the `SyncHdr` element type, the recipient MUST NOT return `Status` commands for any of the commands in the current `SyncML` message.

Example:

```
<Replace CmdID="1" NoStatus="true">
  <Item>
    <SourceClientURI>./127</SourceClientURI>
    <Meta Type="text/directory profile=vCard"/>
    <Data>BEGIN:VCARD
VERSION:2.1
FN:Bruce Smith
N:Smith;Bruce
TEL;TYPE=WORK;VOICE;MSG:+1-919-555-9999
ADR:;;123 Main St.;Anywhere;CA;;US
END:VCARD
    </Data>
  </Item>
</Replace>
```

6.2.49 NumberOfChanges

Usage: Indicates the total number of changes (the number of `Add`, `Replace` and `Delete` commands) that are going to be sent from sender to recipient during a synchronization session so that the recipient MAY use this information to calculate progress information.

Parent Element: `Sync`

Content Model:

```
<xs:attribute name="NumberOfChanges" type="xs:unsignedInt"/>
```

Restrictions:

The attribute MUST be specified by the server, but only if the client has indicated that it supports *`NumberOfChanges`*. It MAY be specified by the client. If synchronizations are carried out on more than one datastore (e.g. Contacts & Calendar), then *`NumberOfChanges`* MUST be specified for each datastore.

Example:

```
<Sync CmdID="5" NumberOfChanges="20">
  ...
  <TargetServerURI>contacts</TargetServerURI>
```

```
  <SourceClientURI>C:\System\data\Contacts.cdb</SourceClientURI>
  ...
</Sync>
```

6.2.50 Put

Usage: Specifies the SyncML command to transfer data items to a recipient network device or database.

Parent Element: SyncBody

Content Model:

```
<xs:element name="Put" type="CommonOperationType"/>
```

Restrictions:

There is no synchronization state information for data transferred using the Put command. This command MUST NOT be specified within a Sync command.

One or more Item element types MUST be specified. The Item element type specifies the data items to be transferred to the recipient. The TargetClientURI/TargetServerURI and SourceClientURI/SourceServerURI specified within the Item element type SHOULD be an absolute URI.

If the command completed successfully, then the (200) OK exception condition is created by the command.

If the originator's authentication credentials specify a principal with insufficient rights to complete the command, then the (401) Unauthorized exception condition is created by the command. If no authentication credentials were specified, then (407) Authentication required exception condition is created by the command. A suitable challenge can also be returned.

If the Put command did not include the size of the data item to be transferred (i.e., in the Meta element's *Size* attribute), then the (411) Size required exception condition is created by the command.

If the data item to be transferred is too large (e.g., there are restrictions on the size of data items transferred to the recipient), then the (413) Request entity too large exception condition is created by the command.

If the *Size* specified in the Meta element type was too large for the recipient (e.g., the recipient does not have sufficient input buffer for the data), then the (416) Requested size too big exception condition is created by the command.

If the MIME content type or content format for the data item is not supported by the recipient, then the (415) Unsupported MIME content type or content format exception condition is created by the command.

If the recipient device storage is full, then the (420) Device full exception condition is created by the command, and the originator SHOULD NOT attempt to add additional data until the recipient has more free space.

Non-specific errors created by the recipient while attempting to complete the command create the (500) Command failed exception condition.

Example: The following is an example of a Put command used to exchange device information.

```
<Put CmdID="12345">
  <Cred>
    <Meta Type="syncml:auth-md5" Format="b64"/>
    <Data>Zz6EivR3yeaaENcRN6lpAQ==</Data>
  </Cred>
  <Meta Type="application/vnd.syncml-devinf+xml"/>
  <Item>
    <SourceClientURI>./devinf20 </SourceClientURI>
    <Data><![CDATA[
      <DevInf xmlns='syncml:devinf20' Version="2.0">
```

```
          … …
      </DevInf>]]>
    </Data>
  </Item>
</Put>
```

6.2.51 Record

Usage: Specifies a record level filter to be performed on the parent element of the `Filter` element.

Parent Element: `Filter`

Content Model:

```
<xs:element name="Record" type="RecordType"/>
<xs:complexType name="RecordType">
  <xs:sequence>
    <xs:element ref="Item"/>
  </xs:sequence>
</xs:complexType>
```

Restrictions:

The `Record` element MAY appear in the `Filter` element for `SyncAlert` elements. If the `Record` element is present, the `SyncAlert` `TargetServerURI/TargetClientURI` `Meta` Type is used to indicate the content type used in the filter record query and MUST be present. The `Record` `Item` element specifies the filter query itself. The `Record` `Item` `Meta` Type element is used to indicate the filter query grammar. The `Record` `Item` `Data` is used to indicate the filter query itself and MUST be present.

Example: The following is an example of a `Record` element used within a `Filter` element to define the characteristics of the subset of data to be synchronized.

1. The `Meta` *`Type`* value of "syncml:filtertype-cgi" indicates the grammar being used.
2. The `Item` `Data` element constrains the items synchronized to those that fall into the "business" or "personal" group (case insensitive) with the cgi expression "GROUP&iCON;business&OR; GROUP &iCON;personal".

```
<Filter>
  <Record>
    <Item>
      <Meta Type="syncml:filtertype-cgi"/>
      <Data><![CDATA[GROUP&iCON;business&OR;GROUP&iCON;personal]]></Data>
    </Item>
  </Record>
</Filter>
```

6.2.52 Replace

Usage: Specifies the `SyncML` command to replace data.

Parent Elements: `Sync`, `SyncBody`

Content Model:

```
<xs:element name="Replace" type="ReplaceType"/>
<xs:complexType name="ReplaceType">
  <xs:sequence>
    <xs:element ref="Cred" minOccurs="0"/>
```

```
    <xs:element ref="Meta" minOccurs="0"/>
    <xs:element ref="Item" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute ref="CmdID" use="required" />
  <xs:attribute ref="FieldLevel" use="optional"/>
  <xs:attribute ref="NoStatus" use="optional"/>
</xs:complexType>
```

Restrictions:

The Replace command is used to replace data on the recipient. The Replace may be either partial (field-level Replace) or full.

If the specified data item does not exist on a server, then the command from a client MUST be interpreted as an Add command. Servers MUST NOT use the Replace command to create new data items on the client. The Replace command MUST NOT be used to Move data items between folders.

The originator of the command SHOULD only send features/properties of the data item that are supported by the recipient. The device information document of the recipient can contain this information.

If the Replace is used for partial update (field-level Replace), then the *FieldLevel* attribute MUST be used.

If the *FieldLevel* attribute is included in the Replace element, and the value is "true", then:

- The sender MAY include unchanged fields of the item inside the Data element.
- The recipient MUST NOT remove any fields of the item in the database that are not present in the Data element.
- The partial item inside the Data MUST still satisfy the validity rules defined for the content type specified in *Type* attribute.
- The item MUST exist on the recipient.

The scope of the meta-information is limited to the command, and MUST match that of any existing object.

Example 1: A server sending a field level Replace command to a client.

```
<Replace CmdID="901" FieldLevel="true">
  <Meta Type="x-type/x-subtype"/>
  <Item>
    <TargetClientURI>1002</TargetClientURI>
    <Data>
      ...
    </Data>
  </Item>
</Replace>
```

One or more Item element types MUST be specified. The Item element type specifies the data item replaced in the database.

Any TargetClientURI, TargetClientParentURI, SourceClientURI or SourceClientParentURI specified within the Item element type MUST be client LUIDs, as relative to the corresponding TargetServerURI / TargetClientURI or SourceServerURI / SourceClientURI specified in the parent Sync command.

Example 2: A client sending a Replace command to a server.

```
<Replace CmdID="902">
  <Meta Type="x-type/x-subtype"/>
  <Item>
    <SourceClientURI FP="1234">1002</SourceClientURI>
    <Data>
```

```
      ...
    </Data>
  </Item>
</Replace>
```

When synchronizing hierarchical objects, the `Replace` command from a client MUST include parent information. For this purpose a `SourceClientParentURI` element MUST be used by the client device referring to an existing parent by LUID. The `Replace` command from a server MUST NOT include parent information.

In the case where an item has been modified and moved, `Replace` MUST NOT be used to `Move` the item.

If the `Replace` command is used by the client for a hierarchical `Add` operation, then the client MUST add the parent first before adding any children of that parent. For example, if an item B has a parent A, then before adding item B, parent A MUST be added by the client.

Example 3: The client sends a hierarchical replace command to the server.

```
<Replace CmdID="903">
  <Item>
    <SourceClientURI FP="2345">1002</SourceClientURI>
    <SourceClientParentURI>1004</SourceClientParentURI>
    <Data>
      ...
    </Data>
  </Item>
</Replace>
```

Status Codes:

If the command completed successfully, then the `(200) OK` exception condition is created by the command. However, if the command was interpreted as an `Add` command and the command completed successfully, then the `(201) Item added` exception condition is created by this command.

If the originator's authentication credentials specify a principal with insufficient rights to complete the command, then the `(401) Unauthorized` exception condition is created by the command.

If no authentication credentials were specified, then `(407) Authentication required` exception condition is created by the command. A suitable challenge can also be returned.

Non-specific errors created by the recipient while attempting to complete the command create the `(500) Command failed` exception condition.

If there is insufficient space on the recipient database for updating the data item, then the `(420) Device full` exception condition is created by the command, and the originator SHOULD NOT attempt to add additional data until the recipient has more free space.

If the MIME content type or content format for the data item is not supported by the recipient, then the `(415) Unsupported media type or format` exception condition is created by the command.

In case the recipient is unable to process a partial item update (e.g. when the item does not exist on recipient), it MUST return `(426) Partial item` not accepted.

Example 4: The following example specifies a server sending a hierarchical `Replace` command to the client. The `TargetClientURI` contains the relative URI of the item that was replaced. The absolute URI of the Source is specified in the parent `Sync` element type (not shown in the example).

```
<Replace CmdID="904">
  <Meta Type="..."/>
  <Item>
```

```
    <TargetClientURI>1002</TargetClientURI>
    <Data>
     ...
    </Data>
  </Item>
</Replace>
```

6.2.53 RespURI

Usage: Specifies the URI that the recipient MUST use for any response to this message.

Parent Element: `SyncHdr`

Content Model:

```
<xs:element name="RespURI" type="xs:anyURI"/>
```

Restrictions:

The value of this element is the address, in the form of an absolute URI that the recipient MUST use for any response to this message. If the Source is not the same as this value, then the `SourceServerURI` element MUST also be specified in the `SyncHdr` element type. Note that the server and databases are the same entities at this new address. The recipient of this command SHOULD NOT resend the previous message.

6.2.54 Results

Usage: Specifies the `SyncML` command that is used to return the results of a `Get` command.

Parent Element: `SyncBody`

Content Model:

```
<xs:element name="Results" type="ResultsType"/>
<xs:complexType name="ResultsType">
  <xs:sequence>
    <xs:element ref="Meta" minOccurs="0"/>
    <xs:element ref="Item" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute ref="CmdID" use="required"/>
  <xs:attribute ref="MsgRef" use="required"/>
  <xs:attribute ref="CmdRef" use="required"/>
</xs:complexType>
```

Restrictions:

The OPTIONAL *MsgRef* attribute specifies the *MsgID* of the associated `SyncML` request. If the *MsgRef* is not present in a `Results` element type, then the *MsgRef* value of "1" MUST be assumed.

The *CmdRef* attribute specifies the *CmdID* of the associated `SyncML` request.

One or more `Item` element types MUST be specified. The `Item` element type specifies the results. The `SourceClientURI/SourceServerURI` specified within the `Item` element type SHOULD be a relative URI, as relative to the corresponding `SourceClientURI/SourceServerURI` in the `Get` command.

Example: The following is an example of `Results` returned from a `Get` command.

```
<Results CmdID="4321" MsgRef="1" CmdRef="1">
  <Meta Type="text/x-vCard"/>
  <Item>
    <SourceClientURI>./1</SourceClientURI>
```

```
    <Data>BEGIN:VCARD
VERSION:2.1
FN:Bruce Smith
N:Smith, Bruce
TEL;WORK;VOICE:+1-919-555-1234
END:VCARD
    </Data>
  </Item>
  <Item>
    <SourceClientURI>./2</SourceClientURI>
    <Data>BEGIN:VCARD
VERSION:2.1
FN:Ida Blue
N:Blue, Ida
TEL;WORK;VOICE:+1-919-555-2345
END:VCARD
    </Data>
  </Item>
  <Item>
    <SourceClientURI>./3</SourceClientURI>
    <Data>BEGIN:VCARD
VERSION:2.1
Mike McGrath
N:McGrath, Mike
TEL;WORK;VOICE:+1-919-555-3456
END:VCARD
    </Data>
  </Item>
</Results>
```

6.2.55 Sequence

Usage: Specifies the `SyncML` attribute to order the processing of a set of `SyncML` commands.

Parent Element: `Sync`

Content Model:

```
<xs:attribute name="Sequence" type="xs:boolean" default="false"/>
```

Restrictions:

If the command with *`Sequence`* attribute completed successfully, then the (`200`) `OK` exception condition is created by the command.

If the recipient does not support the *`Sequence`* attribute, then the (`406`) `Optional feature not supported` exception condition is created by the command.

Non-specific errors created by the recipient while attempting to complete the command with *`Sequence`* attribute create the (`500`) `Command failed` exception condition.

Example: The following is an incomplete (i.e., `Add` and `Delete` commands only include skeleton content) example for a `Sequence` command containing two `Add` commands, followed by a `Delete` command.

```
<Sync Sequence="true" CmdID="1234" >
  <Add CmdID="1235" >
    ...
  </Add>
  <Add CmdID="1236" >
    ...
  </Add>
```

```
  <Delete CmdID="1237" >
    ...
  </Delete>
</Sync>
```

6.2.56 ServerURI

Usage: Specifies the SourceServerURI or TargetServerURI referenced by a `Status` or `Results` element type

Parent Elements: `StatusItem`, `Status`, `Results`

Content Model:

```
<xs:element name="ServerURI" type="xs:anyURI"/>
```

Restrictions:

When specified in the `Status` element type, specifies the Server address specified in the command associated with the response status. When specified in the `StatusItem` element type, specifies the server address specified in the specific `Item`. When specified in the `Results` element type, specifies the Server address specified in the associated `Get` command.

The element type MUST be specified in a `Status` command corresponding to any `SyncML` command that includes the `SourceServerURI` or `TargetServerURI` element type.

Example:

```
<Status CmdID="4321" MsgRef="1" CmdRef="1234" Cmd="Add" Code="200">
  <ServerURI>ABC012345_1012</ServerURI>
  <ClientURI>1012</ClientURI>
</Status>
```

6.2.57 SessionID

Usage: Specifies the identifier of the `SyncML` session associated with the `SyncML` Message.

Parent Element: `SyncHdr`

Content Model:

```
<xs:attribute name="SessionID" type="xs:unsignedInt"/>
```

Restrictions:

The value is an opaque string. The element type MUST be specified in the `SyncHdr` element type in all `SyncML` Messages. The initiator SHOULD use unique *`SessionID`*s for each session.

The maximum length of a *`SessionID`* is 4 bytes. Note for a client having an 8 bit incrementing *`SessionID`* counter is enough for practical implementations.

Example:

```
<SyncML ...>
  <SyncHdr SessionID="1" MsgID="3" >
   ...
  </SyncHdr>
  <SyncBody>
     ...blah, blah...
  </SyncBody>
</SyncML>
```

6.2.58 SftDel

Usage: Indicates that the `Delete` command is a "Soft Delete".

Parent Element: `Delete`

Content Model:

```
<xs:attribute name="SftDel" type="xs:boolean" default="false"/>
```

Restrictions:

The data item is deleted from the client data store but not from the set of synchronization data. The "Soft Delete" can be specified by an OMA DS server to free up storage resources in the OMA DS client prior to a synchronization operation. If not present, then the semantics of the `Delete` command are a "Hard Delete" of the data item. In addition, the OMA DS client can specify the "Soft Delete" to free up storage resources in the OMA DS client prior to a synchronization operation with the OMA DS server.

The OMA DS client MUST maintain the LUID (Local Unique Identifier) associated with the soft-deleted item so that server(s) can re-use the LUID if the item is modified by a server.

The OMA DS server MUST NOT delete the map items associated with the "Soft Deleted" items.

If the OMA DS client does not support the "Soft Delete", then, a (`406`) `Optional feature not supported` MUST be returned in the `Status` command.

In a two-way synchronization, if the OMA DS client specifies a "Soft Delete" for an item that has already been "Hard Deleted" on the OMA DS server, then a (`423`) `Soft-delete conflict` MUST be returned in the `Status` command.

Example: The server requests the client to Soft Delete a set of items to free storage on the device. The items are identified by client LUIDs.

```
<Delete CmdID="403" SftDel="true">
  <Item><TargetClientURI>1018</TargetClientURI></Item>
  <Item><TargetClientURI>1019</TargetClientURI></Item>
  <Item><TargetClientURI>1020</TargetClientURI></Item>
</Delete>
```

6.2.59 Size

Usage: Specifies the byte size of a data object.

Parent Element: `Meta`

Content Model:

```
<xs:attribute name="Size" type="xs:unsignedLong"/>
```

Restrictions:

The byte size is specified as the numeric text equivalent of the byte count of the data object. In case a `Meta` element containing a `Size` attribute contains meta-information about a `Data` object, this `Meta` element MUST have the same parent as the `Data` object it refers to.

Example: The following example illustrates how this attribute is used to specify meta-information about the byte size of the `Item` element type.

```
<Item>
  <TargetClientURI>4</TargetClientURI>
```

```
  <Meta Format="chr" Type="text/plain" Size="10" />
  <Data>John Smith</Data>
</Item>
```

6.2.60 SourceClientURI

Usage: Specifies source routing or mapping information by client identifier.

Parent Elements: `Item`, `Sync`, `SyncHdr`

Content Model:

```
<xs:element name="SourceClientURI" type="URIWithFPType"/>
<xs:complexType name="URIWithFPType">
  <xs:simpleContent>
    <xs:extension base="xs:anyURI">
      <xs:attribute ref="FP" use="optional"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
```

Restrictions:

When specified in the `Item` element type, the `SourceClientURI` element type specifies the client identifier for the item that is the source of the `SyncML` command.

When specified in the `SyncHdr` element type, the `SourceClientURI` element type specifies the source routing information for the client device that originated the `SyncML` Message. This MUST be specified by the client of the session, and MUST NOT be specified by the server of the session.

If the `RespURI` element type is also specified within the `SyncHdr`, then the `SourceClientURI` element type specifies the source routing information for a proxy originator of the `SyncML` message.

When specified in the `Sync` element type, the `SourceClientURI` element type specifies the source routing information of the client datastore originating the data synchronization request.

Example: The following is an example of the usage in an `Item` element type sent from the client.

```
<Replace CmdID="4567" >
  <Item>
    <TargetServerURI>./bruce1/pnab</TargetServerURI>
    <SourceClientURI FP="3456">./contacts</SourceClientURI>
    ... ...
  </Item>
</Replace>
```

6.2.61 SourceClientParentURI

Usage: Specifies the client's parent information of the current item. This may be in the form of a path or the actual Client unique identifier.

Parent Element: `Item`

Content Model:

```
<xs:element name="SourceClientParentURI" type="xs:anyURI"/>
```

Restrictions:

SourceClientParentURI provides parent information of the child that is the current Item of Sync commands such as Move, Add and Replace. SourceClientParentURI MUST be specified in Add, Replace or Move, if and only if the objects have hierarchical nature, i.e. have a parent and child relation, and the client side ID of the parent is known. SourceClientParentURI has meaning only when synchronizing objects in a datastore with hierarchical structure.

In case the parent container is root then the value of the SourceClientParentURI MUST be indicated by '/', without the quotes.

The SourceClientParentURI element represents the client side's ID of an item.

6.2.62 SourceServerURI

Usage: Specifies source routing or mapping information by server identifier.

Parent Element: Item, Sync, SyncHdr

Content Model:

```
<xs:element name="SourceServerURI" type="xs:anyURI"/>
```

Restrictions:

When specified in the Item element type, the SourceServerURI element type specifies the server database item that is the source of the SyncML command.

When specified in the SyncHdr element type, the SourceServerURI element type specifies the source routing information for the network device that originated the SyncML Message. This MUST be specified by the server of the session, and MUST NOT be specified by the client of the session.

If the RespURI element type is also specified within the SyncHdr, then the SourceServerURI element type specifies the source routing information for a proxy originator of the SyncML message.

When specified in the Sync element type, the SourceServerURI element type specifies the source routing information of the server datastore originating the data synchronization request.

Example: The following is an example of the usage in an Item element type sent from the client.

```
<Replace CmdID="4567" >
  <Item>
    <TargetServerURI>./bruce1/pnab</TargetServerURI>
    <SourceClientURI>./contacts</SourceClientURI>
    ... ...
  </Item>
</Replace>
```

6.2.63 SourceServerParentURI

Usage: Specifies the server's parent information of the current item. This may be in the form of a path or the actual Server unique identifier.

Parent Element: Item

Content Model:

```
<xs:element name="SourceServerParentURI" type="xs:anyURI"/>
```

Restrictions:

SourceServerParentURI provides the server's parent information of the current Item of the Sync commands such as Move, Add and Replace. SourceServerParentURI MUST be specified in Add, Replace or Move, if and only if the objects have hierarchical nature, i.e. have a parent and child relation, and only the server side ID of the parent is known. SourceServerParentURI has meaning only when synchronizing objects in a datastore with hierarchical structure.

In case the parent container is root then the value of the SourceServerParentURI MUST be indicated by '/', without the quotes.

The SourceServerParentURI element represents a temporary ID (GUID) of the item that was previously sent by the server to the client and for which the mapping was not yet received by the server.

In the case of moving a child on the server that was already synced with the client to a new parent, which hasn't been synced, the server MUST use SourceServerParentURI. In such situations the new parent has to be added before the Move can be performed. This scenario is further illustrated by the example.

Example:

```
<Add CmdID="12345" >
  <Item>
    <SourceServerURI>1002345/</SourceServerURI>
    <Data>
    ...
    </Data>
  </Item>
</Add>
<!-- Since this is an add from the server to the client, client will
assign an id on its own to the folder added by the server. The mapping of
the client's id with server TempGUID '1002345'  MUST be maintained by the
client till the end of Package 4 -->
<Move CmdID="1234" >
  <Meta Type="text/plain"/>
  <Item>
    <TargetServerURI>110</TargetServerURI>
    <SourceServerParentURI>1002345</SourceServerParentURI>
  </Item>
</Move>
<!-- Since server hasn't received the mapping yet, server will use the
same TempGUID '1002345' when addressing the NEW Parent folder in the
SourceParent of the Move Command -->
```

6.2.64 Status

Usage: Specifies the request status code for a corresponding SyncML command.

Parent Element: SyncBody

Content Model:

```
<xs:element name="Status" type="StatusType"/>
<xs:complexType name="StatusType">
  <xs:sequence>
    <xs:element ref="ServerURI" minOccurs="0"/>
    <xs:element ref="ClientURI" minOccurs="0"/>
    <xs:element ref="Anchor" minOccurs="0"/>
    <xs:element ref="Cred" minOccurs="0"/>
    <xs:element ref="Chal" minOccurs="0"/>
    <xs:element ref="StatusItem" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element ref="Item" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute ref="CmdID" use="required"/>
```

```
  <xs:attribute ref="MsgRef" use="required"/>
  <xs:attribute ref="CmdRef" use="required"/>
  <xs:attribute ref="Cmd" use="required"/>
  <xs:attribute ref="Code"/>
</xs:complexType>
```

Restrictions:

A Status command only applies to the command corresponding to the specified *CmdRef* (i.e., 1:1 correspondence of a command and a Status).

If there were no Item elements specified in the original command, then the *Code* of the Status command MUST contain the status code for the original command. In this case, the ServerURI of the Status command MUST contain the contents of the TargetServerURI of the original command, if one was specified. If there was no TargetServerURI specified in the original command, then the ServerURI of the Status command MUST contain the contents of the SourceServerURI of the original command, if one was specified. If neither were specified, the ServerURI MUST be omitted. Additionally, the ClientURI MUST contain the contents of the TargetClientURI of the original command, if one was specified. If there was no TargetClientURI specified in the original command, then the ClientURI of the Status command MUST contain the contents of the SourceClientURI of the original command, if one was specified. If neither were specified, the ClientURI MUST be omitted.

If there were one or more Item elements specified in the original command, and if the Items' status *Code*s were not the same (if there was more than one), or a client wishes to specify a new identifier for an item then there MUST be a corresponding StatusItem returned for each of the Items of the original command. If this does not apply, StatusItems MAY be returned for each of the Items of the original command. If any StatusItems are returned, there MUST be a StatusItem for each of the Items of the original command.

If there were one or more Item elements specified in the original command and StatusItems are not included in the Status command, the *Code* of the Status command MUST contain the status code for all Item of the original command. Additionally, the ServerURI and ClientURI MUST be omitted.

If StatusItems are included in the Status command, a default status *Code* MAY be set as the *Code* of the Status command. If no default status is specified, each StatusItem MUST include a status *Code*. Any status *Code* specified in a StatusItem overrides any default value. The ServerURI and ClientURI of the Status command MUST be omitted. Additionally, the ServerURI of the StatusItem MUST contain the contents of the TargetServerURI of the original corresponding Item, if a TargetServerURI was specified. If there was no TargetServerURI specified in the corresponding Item, then the ServerURI of the Status command MUST contain the contents of the SourceServerURI of the corresponding Item, if one was specified. If neither were specified, the ServerURI MUST be omitted from that StatusItem. Additionally, the ClientURI of the StatusItem MUST contain the contents of the TargetClientURI of the corresponding Item, if one was specified. If there was no TargetClientURI specified in the corresponding Item, then the ClientURI of the Status command MUST contain the contents of the SourceClientURI of the corresponding Item, if one was specified. If neither were specified, the ClientURI MUST be omitted from the StatusItem.

If the Status command is associated with a command that had other commands inside it (e.g., Sync), then the status value only applies to the corresponding command, and is not related to the status of the commands inside it.

Ordering of Status commands in a SyncML response MUST match the order of the commands in the corresponding SyncML request, except for the Status of the SyncHdr. That is, when there are multiple commands in a SyncML request, then the corresponding Status commands MUST appear in the SyncML response in the same order as the associated commands appeared in the SyncML request. Note that for nested commands, such as an Add command within a Sync command, this means that the Status of the Sync command MUST come before the Status of the Add command.

In addition, the Status on the SyncHdr MUST be the first Status element in the SyncBody of the response. Even in the case where the Statuses for the previous request span multiple messages/responses, the Status on SyncHdr MUST be the first Status element followed by any remaining Statuses from previous messages/responses (maintaining order), followed by any remaining Statuses from the most recent message/response.

The *MsgRef* attribute specifies the *MsgID* of the associated SyncML request.

The *CmdRef* attribute specifies the *CmdID* of the associated SyncML request. The attribute MUST be present. If "0", the Status command corresponds to a status code for the SyncHdr of the SyncML message referenced by the Status command.

The Cmd attribute specifies the name of the SyncML command associated with the SyncML request. The value of this attribute can also be "SyncHdr" when the *CmdRef* attribute has a value of "0".

The Chal element type specifies the authentication challenge for the command or the message. If the status code in the *Code* attribute is (401) Unauthorized or (407) Authentication required, the challenge SHOULD be included. If the status code in the *Code* attribute is (441) Encrypted symmetry key error or (447) Encryption required, the challenge SHOULD be included.

The OPTIONAL Anchor attribute specifies the sync anchor information.

The *Code* attribute specifies the request status code type.

The OPTIONAL and repeatable StatusItem element type contains additional information about the status of individual Items of the original command. The StatusItem element may specify changes to the server's item identifier map table. Map tables are used to correlate small resolution item identifiers with larger resolution item identifiers, or between two differing name spaces of identifiers. The client identifier for a given item is known as a LUID. The server identifier for a given item is known as a GUID. For example, if a mobile device has 2-byte item identifiers (LUIDs) and a network server has 16-byte item identifiers (GUIDs), a map table is necessary to correlate an equivalent 2-byte and 16-byte identifier. Generally, map tables are maintained by the data synchronization engine on the network server. Item identifier map tables are not necessary when the originator and the recipient databases are exact replicas of each other (i.e., the databases have the same physical schema).

If an item identifier map table is needed, it is the responsibility of the recipient/server maintaining the map table to perform item identifier translations when communicating synchronization commands with the client that required the map table.

The processing of a Status command that contains StatusItem elements that provide mapping MUST be atomic in nature. This means that the recipient/server MUST process either the entire list of mappings supplied, or none of them. If the operation fails, the recipient/server SHOULD request a recovery sync when appropriate.

This specification permits a Status command to be issued against another Status command. This case will probably not normally be encountered. However, there are extreme cases where this feature is necessary. For example, if a server returns a (401) Unauthorized status code with a request for an authentication scheme that is not supported by the client, the client might use a (406) Optional feature unsupported to notify the server that that requested authentication scheme is not supported and negotiate an authentication scheme it does support. SyncML servers and SyncML clients not supporting such a usage case need provide no further response to the SyncML entity issuing the "Status on a Status".

The OPTIONAL Item element type MAY contain additional information relevant to the original command.

A Status MUST also be returned for the SyncHdr. However, if a client creates a message containing only a successful Status on a SyncHdr, the entire message MUST NOT be sent. A server MUST send this message.

Status codes are listed in Section 10, Response Status Codes.

Example:

```
<SyncBody>
  <Status CmdID="8765" MsgRef="1" CmdRef="1234" Cmd="Add" Code="401">
    <ServerURI></ServerURI>
    <ClientURI>IMEI:001004FF1234567</ClientURI>
  </Status>
</SyncBody>
```

6.2.65 StatusItem

Usage: Specifies the Item receiving a status, as well as an optional ID mapping.

Parent Element: `Status`

Content Model:

```
<xs:element name="StatusItem" type="StatusItemType"/>
<xs:complexType name="StatusItemType">
  <xs:sequence>
    <xs:element ref="ServerURI" minOccurs="0"/>
    <xs:element ref="ClientURI" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute ref="Code" use="optional" />
</xs:complexType>
```

Restrictions:

The `ClientURI` element type specifies the relative URI for the client's item identifier (LUID). This is either an existing identifier, taken from the `SourceClientURI` or `TargetClientURI` of the associated `Item`, or a new client identifier may be specified by the client if one did not previously exist.

The `ServerURI` element type specifies server's item identifier (GUID) as a relative URI taken from the `SourceServerURI` or `TargetServerURI` of the associated `Item`, if one was specified.

Example:

```
<StatusItem>
  <ServerURI>ABC012345_100x</ServerURI>
  <ClientURI>100x</ClientURI>
</StatusItem>
```

6.2.66 Sync

Usage: Specifies the `SyncML` command that indicates a data synchronization operation.

Parent Element: `SyncBody`

Content Model:

```
<xs:element name="Sync" type="SyncCmdType"/>
<xs:complexType name="SyncCmdType">
  <xs:sequence>
    <xs:element ref="Cred" minOccurs="0"/>
    <xs:choice>
      <xs:element ref="TargetClientURI" minOccurs="0"/>
      <xs:element ref="TargetServerURI" minOccurs="0"/>
    </xs:choice>
    <xs:choice>
      <xs:element ref="SourceClientURI" minOccurs="0"/>
      <xs:element ref="SourceServerURI" minOccurs="0"/>
    </xs:choice>
    <xs:element ref="Meta" minOccurs="0"/>
    <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element ref="Add"/>
      <xs:element ref="Copy"/>
      <xs:element ref="Delete"/>
      <xs:element ref="Move"/>
      <xs:element ref="Replace"/>
    </xs:choice>
  </xs:sequence>
  <xs:attribute ref="CmdID" use="required"/>
  <xs:attribute ref="MaxObjSize" use="optional"/>
  <xs:attribute ref="Atomic" use="optional"/>
```

```
  <xs:attribute ref="Sequence" use="optional"/>
  <xs:attribute ref="NumberOfChanges" use="optional"/>
  <xs:attribute ref="FreeID" use="optional"/>
  <xs:attribute ref="FreeMem" use="optional"/>
  <xs:attribute ref="NoStatus" use="optional"/>
</xs:complexType>
```

Restrictions:

The `TargetClientURI` / `TargetServerURI` element type MUST be used to specify the recipient database to be synchronized.

The `SourceClientURI` / `SourceServerURI` element type MUST be used to specify the originator database to be synchronized.

The optional datastore-specific `FreeID` and `FreeMem` attributes in the `Sync` command MUST be associated with the source datastore specified in the `SourceClientURI` / `SourceServerURI` element of the `Sync` command.

Zero or more `Add`, `Replace`, `Delete` `Copy` or `Move` element types MUST be specified. There is no implied order to the processing of these commands unless the *`Sequence`* attribute is set.

If the command completed successfully, then the (`200`) `OK` exception condition is created by the command.

If the originator's authentication credentials specify a principal with insufficient rights to complete the command, then the (`401`) `Unauthorized` exception condition is created by the command. If no authentication credentials were specified, then (`407`) `Authentication required` exception condition is created by the command. A suitable challenge can also be returned.

If the originator's authentication credentials specify a principal that has had its rights to issue `Sync` commands denied, then the (`403`) `Forbidden` exception condition is created by this command. However, if the recipient does not want to make this fact public, then the (`404`) `Not found` exception condition can be used.

If the recipient does not allow `Sync` commands either on the specified database or on the network device, then the (`405`) `Command not allowed` exception condition is created by this command.

If the specified database cannot be found on the recipient network device, then the (`404`) `Not found` exception condition is created by this command.

If the recipient determines that there is a high probability that the client device data is out of sync, then the (`508`) `Refresh required` exception condition is created by this command. When this exception condition occurs, the originator of the `Sync` command SHOULD initiate a slow synchronization with the recipient.

Non-specific errors created by the recipient while attempting to complete the command create the (`500`) `Command failed` exception condition.

Example: The following is an example of a `Sync` command with authentication credentials and a single `Add` of a calendar entry.

```
<Sync CmdID="1234">
  <Cred>
    <Meta Type="syncml:auth-md5" Format="b64"/>
    <Data>Zz6EivR3yeaaENcRN6lpAQ==</Data>
  </Cred>
  <TargetServerURI>./mail/bruce1</TargetServerURI>
  <SourceClientURI>./calendar</SourceClientURI>
  <Add CmdID="1246">
    <Item>
      <SourceClientURI>./12</SourceClientURI>
      <Meta Type="text/x-vCalendar"/>
      <Data>BEGIN:VCALENDAR
VERSION:1.0
BEGIN:VEVENT
DTSTART:20000531T160000Z
```

```
DTEND:20000531T160100Z
SUMMARY:Release v0.9 of specs
END:VEVENT
END:VCALENDAR
      </Data>
    </Item>
  </Add>
</Sync>
```

6.2.67 SyncAlert

Usage: Specifies the parameters for the sync type negotiation. Both the data sync client and server can use this element to alert the other side to initiate a specific data synchronization session.

Parent Element: `SyncBody`

Content Model:

```
<xs:element name="SyncAlert" type="SyncAlertType"/>
<xs:complexType name="SyncAlertType">
  <xs:sequence>
    <xs:element ref="Anchor"/>
    <xs:element ref="Cred" minOccurs="0"/>
    <xs:choice>
      <xs:element ref="TargetClientURI"/>
      <xs:element ref="TargetServerURI"/>
    </xs:choice>
    <xs:choice>
      <xs:element ref="SourceClientURI"/>
      <xs:element ref="SourceServerURI"/>
    </xs:choice>
    <xs:element ref="SyncType"/>
    <xs:element ref="IDContainer" minOccurs="0"/>
    <xs:element ref="Filter" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute ref="CmdID" use="required"/>
  <xs:attribute ref="MaxObjSize" use="optional"/>
  <xs:attribute ref="Correlator" use="optional"/>
  <xs:attribute ref="NoStatus" use="optional"/>
</xs:complexType>
```

Restrictions:

The `SyncType` element specifies the parameters for the sync type negotiation.

The *`IDValidity`* attribute specifies if the ID is valid.

The semantics of *`Direction`*, *`Behaviour`*, *`IDValidity`* and *`ChangeLogValidity`* are described in [DSPRO].

The `TargetClientURI` / `TargetServerURI` and `SourceClientURI` / `SourceServerURI` elements specify the target and source address to be synchronized. The semantics of `TargetClientURI` / `TargetServerURI` and `SourceClientURI` / `SourceServerURI` are described in [DSPRO].

The optional `Filter` element specifies the filter criteria.

The optional `IDContainer` element specifies one or more data item identifiers.

The *`Last`* and *`Next`* attributes of the `Anchor` element specify the synchronization state information for the last and current session, respectively.

If multiple target and source addresses need to be synchronized within one session, the Data Sync Client or Server MUST use the `SyncAlert` command for each target and source pair to negotiate the sync types.

If the command and the associated action are completed successfully, then the status code '(200) OK' is created by the command.

Example: The following is an example for a data sync client to initiate a normal sync from client to server only.

```
<SyncAlert CmdID="1">
  <Anchor Last="234" Next="276"/>
  <Cred>
    <Meta Type="syncml:auth-sha256" Format="b64"/>
    <Data>...</Data>
  </Cred>
  <TargetServerURI>/Macy/02</TargetServerURI>
  <SourceClientURI>/Macy/02</SourceClientURI>
  <SyncType Direction="fromClient" Behaviour="Preserve" IDValidity="true"
    ChangeLogValidity="true"/>
</SyncAlert>
```

6.2.68 SyncBody

Usage: Specifies the container for the body or contents of the `SyncML` message.

Parent Element: `SyncML`

Content Model:

```
<xs:element name="SyncBody" type="SyncBodyType"/>
<xs:complexType name="SyncBodyType">
  <xs:sequence>
    <xs:choice maxOccurs="unbounded">
      <xs:element ref="Add"/>
      <xs:element ref="Alert"/>
      <xs:element ref="Copy"/>
      <xs:element ref="Delete"/>
      <xs:element ref="Get"/>
      <xs:element ref="Move"/>
      <xs:element ref="Put"/>
      <xs:element ref="Replace"/>
      <xs:element ref="Results"/>
      <xs:element ref="Status"/>
      <xs:element ref="Sync"/>
      <xs:element ref="SyncAlert"/>
    </xs:choice>
    <xs:element ref="Final" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

Restrictions:

None.

Example:

```
<SyncML xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation=
    "OMA-SUP-XSD_DS_Syntax_Schema-V2_0.xsd"
    Version="2.0" >
  <SyncHdr SessionID="1" MsgID="1">
    <TargetClientURI>IMEI:001004FF1234567</TargetClientURI>
<SourceServerURI>http://www.datasync.org/servlet/syncit</SourceServerURI>
    <Cred AuthName="Bruce">
      <Meta Type="syncml:auth-md5" Format="b64"/>
      <Data>Zz6EivR3yeaaENcRN6lpAQ==</Data>
```

```
    </Cred>
  </SyncHdr>
  <SyncBody>
    <Get CmdID="1234">
      <Item>
        <TargetClientURI>./devinf20</TargetClientURI>
      </Item>
    </Get>
  </SyncBody>
</SyncML>
```

6.2.69 SyncHdr

Usage: Specifies the container for the provisioning, routing information in the `SyncML` message.

Parent Element: `SyncML`

Content Model:

```
<xs:element name="SyncHdr" type="SyncHdrType"/>
<xs:complexType name="SyncHdrType">
  <xs:sequence>
    <xs:choice>
      <xs:element ref="TargetClientURI"/>
      <xs:element ref="TargetServerURI"/>
    </xs:choice>
    <xs:choice>
      <xs:element ref="SourceClientURI"/>
      <xs:element ref="SourceServerURI"/>
    </xs:choice>
    <xs:element ref="RespURI" minOccurs="0"/>
    <xs:element ref="Cred" minOccurs="0"/>
    <xs:element ref="Chal" minOccurs="0"/>
    <xs:element ref="EncryptedKey" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute ref="SessionID" use="required"/>
  <xs:attribute ref="MsgID" use="required"/>
  <xs:attribute ref="MaxMsgSize" use="optional"/>
  <xs:attribute ref="NoStatus" use="optional"/>
</xs:complexType>
```

Restrictions:

The OPTIONAL *MaxMsgSize* attribute is used to convey the maximum byte size of a `SyncML` response.

Example:

```
<SyncML xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation=
    "OMA-SUP-XSD_DS_Syntax_Schema-V2_0.xsd"
    Version="2.0" >
  <SyncHdr SessionID="1" MsgID="1">
    <TargetServerURI>
      http://www.datasync.org/servlet/syncit
    </TargetServerURI>
    <SourceClientURI>IMEI:001004FF1234567</SourceClientURI>
  </SyncHdr>
  <SyncBody>
    ...blah, blah...
  </SyncBody>
</SyncML>
```

6.2.70 SyncML

Usage: Specifies the container for a `SyncML` Message.

Parent Elements: None. This is the root or document element.

Content Model:

```
<xs:element name="SyncML">
  <xs:annotation>
    <xs:documentation>Root</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="SyncHdr"/>
      <xs:element ref="SyncBody"/>
    </xs:sequence>
    <xs:attribute ref="Version" use="required"/>
  </xs:complexType>
</xs:element>
```

Restrictions:

Within transports that support MIME content-type identification, this object MUST be identified as `application/vnd.syncml+xml` (for clear-text, XML representation) or `application/vnd.syncml+wbxml` (for binary, WBXML representation).

Example:

```
<SyncML xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation=
    "OMA-SUP-XSD_DS_Syntax_Schema-V2_0.xsd"
    Version="2.0" >
  <SyncHdr SessionID="1" MsgID="1">
    ...
  </SyncHdr>
  <SyncBody>
    ...blah, blah...
  </SyncBody>
</SyncML>
```

6.2.71 SyncType

Usage: Specifies the parameters for the sync type negotiation.

Parent Element: `SyncAlert`

Content Model:

```
<xs:element name="SyncType" type="SyncTypeType"/>
<xs:complexType name="SyncTypeType" >
  <xs:attribute ref="Direction" use="required" />
  <xs:attribute ref="Behaviour" use="required"/>
  <xs:attribute ref="IDValidity" use="optional" default="true"/>
  <xs:attribute ref="ChangeLogValidity" use="optional" default="true"/>
</xs:complexType>
```

Restrictions:

The SyncType element specifies the parameters for the sync type negotiation.

6.2.72 TargetClientURI

Usage: Specifies target routing or mapping information by client identifier.

Parent Elements: Item, Sync, SyncHdr

Content Model:

```
<xs:element name="TargetClientURI" type="xs:anyURI"/>
```

Restrictions:

When specified in the Item element type, the TargetClientURI element type specifies the server identifier for the item that is the target of the SyncML command.

When specified in the SyncHdr element type, the TargetClientURI element type specifies the target routing information for the client device that is receiving the SyncML Message. This MUST be specified by the server of the session, and MUST NOT be specified by the client of the session.

When specified in the Sync element type, the TargetClientURI element type specifies the source routing information of the database receiving the data synchronization request.

6.2.73 TargetClientParentURI

Usage Specifies the client's parent information of the current Item. This may be in the form of a path or the actual Client unique identifier.

Parent Element: Item

Content Model:

```
<xs:element name="TargetClientParentURI" type="xs:anyURI"/>
```

Restrictions:

TargetClientParentURI provides parent information of the child that is the current Item of Sync commands such as Add, Move and Replace. TargetClientParentURI MUST be specified in Add, Replace and Move, if and only if the objects have hierarchical nature, i.e. have a parent and child relation, and the client side ID of the parent is known. TargetClientParentURI has meaning only when synchronizing objects with hierarchical structure.

In case the parent container is a root then the value of the TargetClientParentURI MUST be indicated by '/', without the quotes.

TargetClientParentURI element always represents a LUID, i.e. the ID of client that was previously sent by the client in a StatusItem element. TargetClientParentURI MUST be sent only by servers.

Example: the server requests the client to move the item having the LUID '110'.

```
<Move CmdID="1234" >
  <Cred>
    <Meta Type="syncml:auth-md5" Format="b64"/>
    <Data>Zz6EivR3yeaaENcRN6lpAQ==</Data>
  </Cred>
  <Meta Type="text/plain"/>
  <Item>
    <TargetClientURI>110</TargetClientURI>
    <TargetClientParentURI>1234</TargetClientParentURI>
```

```
  </Item>
</Move>
```

6.2.74 TargetServerURI

Usage: Specifies target routing or mapping information by server identifier.

Parent Elements: `Item`, `Sync`, `SyncHdr`

Content Model:

```
<xs:element name="TargetServerURI" type="xs:anyURI"/>
```

Restrictions:

When specified in the `Item` element type, the `TargetServerURI` element type specifies the server database item that is the target of the `SyncML` command.

When specified in the `SyncHdr` element type, the `TargetServerURI` element type specifies the target routing information for the network device that receives the `SyncML` Message. This MUST be specified by the client of the session, and MUST NOT be specified by the server of the session.

When specified in the `Sync` element type, the `TargetServerURI` element type specifies the source routing information of the server database receiving the data synchronization request.

6.2.75 TargetServerParentURI

Usage Specifies the parent information of the current `Item`. This may be in the form of a path or the actual Server unique identifier.

Parent Element: `Item`

Content Model:

```
<xs:element name="TargetServerParentURI" type="xs:anyURI"/>
```

Restrictions:

`TargetServerParentURI` provides parent information of the current `Item` of `Sync` commands such as `Add`, `Move` and `Replace`. `TargetServerParentURI` MUST be specified in `Add`, `Replace` and `Move`, if and only if the objects have hierarchical nature, i.e. have a parent and child relation, and the client identifier is not known. `TargetServerParentURI` has meaning only when synchronizing objects with hierarchical structure.

In case the parent container is a root then the value of the `TargetServerParentURI` MUST be indicated by '/', without the quotes.

`TargetServerParentURI` element always represents a GUID. `TargetServerParentURI` MUST be sent only by servers.

Example: the server requests the client to move the item having the LUID '110'.

```
<Move CmdID="1234" >
  <Cred>
    <Meta Type="syncml:auth-md5" Format="b64"/>
    <Data>Zz6EivR3yeaaENcRN6lpAQ==</Data>
  </Cred>
  <Meta Type="text/plain"/>
  <Item>
```

```
    <TargetClientURI>110</TargetClientURI>
    <TargetServerParentURI>1234</TargetServerParentURI>
  </Item>
</Move>
```

6.2.76 Type

Usage: Specifies the media type of the content information in the `Data` element.

Parent Element: `Meta`

Restrictions:

If this attribute is missing, then the default content-type is text/plain. The content information for this attribute SHOULD BE a registered MIME content-type. Alternatively, a URN can be used to specify the media type. In case a `Meta` element containing a `Type` attribute contains meta-information about a `Data` object, this `Meta` element MUST have the same parent as the `Data` object it refers to.

Content Model:

```
<xs:attribute name="Type" type="xs:string"/>
```

Example: The following example illustrates how this attribute is used within a `SyncML` message to specify meta-information about the media type of the content information in the `Item` element type.

```
<Item>
  <TargetClientURI>3</TargetClientURI>
  <Meta Format="chr" Type="text/directory;profile=vCard" />
  <Data>BEGIN:VCARD
VERSION:3.0
FN:Jim Smith
N:Smith;Jim
TEL;TYPE=WORK,VOICE,FAX:+1-919-555-1234
EMAIL;TYPE=INTERNET,WORK:Jim_Smith@mail.host.com
END:VCARD
  </Data>
</Item>
```

6.2.77 Version

Usage: Specifies the major and minor version identifier of the DS Syntax protocol specification used to represent the `SyncML` message.

Parent Element: `SyncML`

Content Model:

```
<xs:attribute name="Version" type="xs:string"/>
```

Restrictions:

Each `SyncML` Message in each SyncML Package sent from an originator to a recipient MUST include the *`Version`* attribute in the `SyncHdr`.

Major revisions of the specification create incompatible changes that will generally require a new `SyncML` parser. Minor revisions involve changes that do not impact basic compatibility of the parser. When the XML document conforms to this revision of the OMA DS Syntax specification the value MUST be `2.0`.

Example:

```
<SyncML xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation=
    "OMA-SUP-XSD_DS_Syntax_Schema-V2_0.xsd"
    Version="2.0">
  <SyncHdr SessionID="1" MsgID="1" >
    ...
  </SyncHdr>
  <SyncBody>
    ...
  </SyncBody>
</SyncML>
```

7. XML Schema

Please refer to the Syntax Schema specification.

8. WBXML Definition

The following tables define the token assignments for the mapping of the SyncML related Schemas and element types into WBXML as defined by [WBXML].

8.1 Code Space Definitions

This version of the SyncML representation protocol specification maps all the SyncML related Schemas into WBXML code spaces.

DTD Name	WBXML PUBLICID Token (Hex Value)	Formal Public Identifier
SyncML	FD1	-//SYNCML//DTD SyncML 1.0//EN
SyncML 1.1	FD3	-//SYNCML//DTD SyncML 1.1//EN
SyncML 1.2	0x1201	-//SYNCML//DTD SyncML 1.2//EN
SyncML 2.0	0x1205	-//SYNCML//Schema SyncML 2.0//EN

Table 10: Code Space Definitions

The SyncML Schema is assigned the WBXML document public identifier (i.e., the "publicid" WBXML BNF production) associated with the 0x1205 token.

8.2 Code Page Definitions

The following code page tokens represent SyncML related public identifiers. This version of the SyncML representation protocol specification utilizes the WBXML code page tokens for identifying schema.

DTD Name	WBXML Code Page Token (Hex Value)	Formal Public Identifier
SyncML	00	-//SYNCML//Schema SyncML 2.0//EN

Table 11: Code Page Definitions

8.3 Element Token Definitions

The following WBXML token codes represent element types (i.e., tags) from code page x00 (zero), SyncML Schema.

Element Type Name	WBXML Tag Token (Hex Value)
Add	05
Alert	06
Anchor	07
ClientURI	08
Chal	09
EncryptedKey	0A
ID	0B
IDContainer	0C
Copy	0D
Cred	0E
Data	0F
Delete	10
NextNonce	11
Final	12
Get	13
Item	14
ServerURI	15
SourceClientURI	16
SourceClientParentURI	17
SourceServerURI	18
SourceServerParentURI	19
Meta	1A
StatusItem	1B
SyncAlert	1C
SyncType	1D
TargetClientURI	1E
Put	1F
Replace	20

Element Type Name	WBXML Tag Token (Hex Value)
RespURI	21
Results	22
TargetClientParentURI	23
TargetServerURI	24
TargetServerParentURI	25
Reserved for future use	26 ~ 28
Status	29
Sync	2A
SyncBody	2B
SyncHdr	2C
SyncML	2D
Reserved for future use	2E ~ 33
MoreData	34
Field	35
Filter	36
Record	37
Reserved for future use	38 ~ 3A
Move	3B
Reserved for future use	3C ~ 3F

Table 12: Element Token Definitions – Tag Order

Element Type Name	WBXML Tag Token (Hex Value)
Add	05
Alert	06
Anchor	07
Chal	09
ClientURI	08
Copy	0D
Cred	0E
Data	0F
Delete	10
EncryptedKey	0A
Field	35
Filter	36
Final	12
Get	13
ID	0B
IDContainer	0C
Item	14
Meta	1A
MoreData	34
Move	3B
NextNonce	11
Put	1F
Record	37
Replace	20
Reserved for future use	26 ~ 28
Reserved for future use	2E ~ 33
Reserved for future use	38 ~ 3A
Reserved for future use	3C ~ 3F
RespURI	21
Results	22

Element Type Name	WBXML Tag Token (Hex Value)
ServerURI	15
SourceClientParentURI	17
SourceClientURI	16
SourceServerParentURI	19
SourceServerURI	18
Status	29
StatusItem	1B
Sync	2A
SyncAlert	1C
SyncBody	2B
SyncHdr	2C
SyncML	2D
SyncType	1D
TargetClientParentURI	23
TargetClientURI	1E
TargetServerParentURI	25
TargetServerURI	24

Table 13: Element Token Definitions - Alphabetical

8.4 Attribute Start Token Definitions

The following WBXML token codes represent the start of an attribute (some including value) in code page x00 (zero), SyncML Schema.

Attribute Name	Attribute Value Prefix	WBXML Tag Token (Hex Value)
Atomic*	false	05
Atomic	true	06
AuthName		07
Behaviour	Preserve	08
Behaviour	Refresh	09
ChangeLogValidity	false	0A
ChangeLogValidity	true	0B
Cmd	Add	0C
Cmd	Alert	0D
Cmd	Copy	0E
Cmd	Delete	0F
Cmd	Get	10
Cmd	Move	11
Cmd	Put	12
Cmd	Replace	13
Cmd	Results	14
Cmd	Status	15
Cmd	Sync	16
Cmd	SyncAlert	17
Cmd	SyncHdr	18
CmdID		19
CmdRef		1A
Code		1B
Code	200	1C
Code	201	1D
Correlator		1E
Direction	fromClient	1F
Direction	fromServer	20

Attribute Name	Attribute Value Prefix	WBXML Tag Token (Hex Value)
Direction	NoWay	21
Direction	twoWay	22
Encrypted*	false	23
Encrypted	true	24
FieldLevel*	false	25
FieldLevel	true	26
FilterType	EXCLUSIVE	27
FilterType	INCLUSIVE	28
Format		29
Format	b64	2A
Format	bin	2B
Format	bool	2C
Format	chr	2D
Format	date	2E
Format	float	2F
Format	int	30
Format	node	31
Format	null	32
Format	time	33
Format	xml	34
FP		35
FreeID		36
FreeMem		37
IDValidity	false	38
IDValidity	true	39
Last		3A
MaxMsgSize		3B
MaxObjSize		3C
MsgID		3D
MsgRef		3E

Attribute Name	Attribute Value Prefix	WBXML Tag Token (Hex Value)
Next		3F
NoStatus*	false	45
NoStatus	true	46
NumberOfChanges		47
Sequence*	false	48
Sequence	true	49
SessionID		4A
SftDel*	false	4B
SftDel	true	4C
Size		4D
Type		4E
Type	application/	4F
Type	application/vnd.omads	50
Type	application/vnd.omads-email+xml	51
Type	application/vnd.omads-file+xml	52
Type	application/vnd.omads-folder+xml	53
Type	application/vnd.syncml-devinf+xml	54
Type	application/vnd.syncml-devinf+wbxml	55
Type	syncml:	56
Type	syncml:auth-sha256	57
Type	text/	58
Type	text/calendar	59
Type	text/directory;profile=vCard	5A
Type	text/plain	5B
Type	text/vcard	5C
Type	text/x-calendar	5D
Type	text/x-vcard	5E
Version		5F
Version	2.0	60
Type	application/vnd.omads-email+wbxml	61

Attribute Name	Attribute Value Prefix	WBXML Tag Token (Hex Value)
Type	application/vnd.omads-file+wbxml	62
Type	application/vnd.omads-folder+wbxml	63
Reserved for future use		64 ~ 7F

Table 14: Attribute Start Token Definitions – Alphabetical

* indicates the default value for the attribute

8.5 Attribute Value Token Definitions

The following WBXML token codes represent attribute values in code page x00 (zero), SyncML Schema.

Attribute Value	WBXML Tag Token (Hex Value)
.com/	85
.edu/	86
.net/	87
.org/	88
http://	89
http://www.	8A
https://	8B
https://www.	8C
xml	8D
wbxml	8E
Reserved for future use	8F ~ BF
Reserved for future use	C5 ~ FF

Table 15: Attribute Value Definitions

9. Common URI Scheme Types

The following is a list of common URI scheme types

URI Scheme Type	Description
FTP	File Transfer Protocol
HTTP	Hypertext Transfer Protocol
IMEI	International Mobile Equipment Identifier
LDAP	Lightweight Directory Access Protocol
OBEX	IrDA Object Exchange Protocol
SYNCML	SyncML specific, as defined in one of the protocol or format specifications
WSP	Wireless Session Protocol
ESN	Electronic Serial Number
MEID	Mobile Equipment Identity

Table 16: Common URI Scheme Types

10.Alert Types

The alert types in SyncML are a numeric text value. The types are divided into two classes, User Alert, that are intended to be conveyed to the recipient's user agent, and Application Alert, that are intended to be conveyed to a target application on the recipient. The only valid values are the standard values defined in this specification.

Implementations that desire to add to these values SHOULD submit a change request to mailto:technical-comments@openmobilealliance.org

Alert Code Value	Name	Description
Alert Codes used for user alerts		
100	DISPLAY	Show. The Data element type contains content information that SHOULD be processed and displayed through the user agent.
101-150	-	Reserved for future SyncML usage.
Special Alert Codes		
211	END SESSION	Specifies a request to end the session.
212	ALERT POLL	Specifies a request to poll.
213	ALERT IDLE	Specifies a request to idle.
214-220	-	Reserved for future SyncML usage.
221	RESULT ALERT	Specifies a request for synchronization results.
222	NEXT MESSAGE	Specifies a request for the next message in the package.
223	NO END OF DATA	End of Data for chunked object not received.
224-250	-	Reserved for future SyncML usage.

Table 17: Alert Types

11.Response Status Codes

The response status codes in SyncML are a numeric text value. The codes are divided into five classes. The only valid values are the standard values defined in this specification.

Implementations that desire to add to these values SHOULD submit a change request to mailto:technical-comments@openmobilealliance.org.

Status Codes	Reason Phrase
Informational 1xx	
101	In progress. The specified SyncML command is being carried out, but has not yet completed.
Successful 2xx	
200	OK. The SyncML command completed successfully.
201	Item added. The requested item was added.
202	Accepted for processing. The request to alert a user or application was successfully performed.
203	Non-authoritative response. The request is being responded to by an entity other than the one targeted. The response is only to be returned when the request would have been resulted in a 200 response code from the authoritative target.
204	No content. The request was successfully completed but no data is being returned. The response code is also returned in response to a Get when the target has no content.
205	Reset content. The source SHOULD update their content. The originator of the request is being told that their content SHOULD be synchronized to get an up to date version.
206	Partial content. The response indicates that only part of the command was completed. If the remainder of the command can be completed later, then when completed another appropriate completion request status code SHOULD be created.
207	Conflict resolved with merge. The response indicates that the request created a conflict; which was resolved with a merge of the client and server instances of the data. The response includes both the target and source URLs in the `Item` of the `Status`. In addition, a Replace command is returned with the merged data.
208	Conflict resolved with client's command "winning". The response indicates that there was an update conflict; which was resolved by the client command winning.
209	Conflict resolved with duplicate. The response indicates that the request created an update conflict; which was resolved with a duplication of the client's data being created in the server database. The response includes both the target URI of the duplicate in the `Item` of the `Status`. In addition, in the case of a two-way synchronization, an `Add` command is returned with the duplicate data definition.
210	Reserved for future SyncML usage.
211	`Item` not deleted. The requested item was not found. It could have been previously deleted.
212	Authentication accepted. No further authentication is needed for the remainder of the synchronization session. This response code can only be used in response to a request in which the credentials were provided.
213	Chunked item accepted and buffered.
214	Operation cancelled. The SyncML command completed successfully, but no more commands will be processed within the session.
215	Not executed. A command was not executed, as a result of user interaction and user chose not to accept the choice.
216	Atomic roll back OK. A command was inside `Atomic` element and `Atomic` failed. This command was rolled back successfully.
Redirection 3xx	
300	Multiple choices. The requested target is one of a number of multiple alternatives requested target. The alternative SHOULD also be returned in the `Item` element type in the `Status`.
301	Moved permanently. The requested target has a new URI. The new URI SHOULD also be returned in the `Item` element type in the `Status`.
302	Found. The requested target has temporarily moved to a different URI. The original URI SHOULD continue to be used. The URI of the temporary location SHOULD also be returned in the `Item` element type in the `Status`. The requestor SHOULD confirm the identity and authority of the temporary URI to act on behalf of the original target URI.
303	See other. The requested target can be found at another URI. The other URI SHOULD be returned in the `Item` element type in the `Status`.
304	Not modified. The requested SyncML command was not executed on the target. This is an additional response that can be added to any of the other Redirection response codes.
305	Use proxy. The requested target MUST be accessed through the specified proxy URI. The proxy URI SHOULD also be returned in the `Item` element type in the `Status`.
Originator Exceptions 4xx	
400	Bad request. The requested command could not be performed because of malformed syntax in the command. The

	malformed command MAY also be returned in the `Item` element type in the `Status`.
401	Invalid credentials. The requested command failed because the requestor MUST provide proper authentication. If the property type of authentication was presented in the original request, then the response code indicates that the requested command has been refused for those credentials.
402	Payment needed. The requested command failed because proper payment is needed. This version of SyncML does not standardize the payment mechanism.
403	Forbidden. The requested command failed, but the recipient understood the requested command. Authentication will not help and the request SHOULD NOT be repeated. If the recipient wishes to make public why the request was denied, then a description MAY be specified in the `Item` element type in the `Status`. If the recipient does not wish to make public why the request was denied then the response code 404 MAY be used instead.
404	Not found. The requested target was not found. No indication is given as to whether this is a temporary or permanent condition. The response code 410 SHOULD be used when the condition is permanent and the recipient wishes to make this fact public. This response code is also used when the recipient does not want to make public the reason for why a requested command is not allowed or when no other response code is appropriate.
405	Command not allowed. The requested command is not allowed on the target. The recipient SHOULD return the allowed command for the target in the `Item` element type in the `Status`.
406	Optional feature not supported. The requested command failed because an OPTIONAL feature in the request was not supported. The unsupported feature SHOULD be specified by the `Item` element type in the `Status`.
407	Missing credentials. This response code is similar to 401 except that the response code indicates that the originator MUST first authenticate with the recipient. The recipient SHOULD also return the suitable challenge in the `Chal` element type in the `Status`.
408	Request timeout. An expected message was not received within the REQUIRED period of time. The request can be repeated at another time. The `RespURI` can be used to specify the URI and optionally the date/time after which the originator can repeat the request. See `RespURI` for details.
409	Conflict. The requested failed because of an update conflict between the client and server versions of the data. Setting of the conflict resolution policy is outside the scope of this version of SyncML. However, identification of conflict resolution performed, if any, is within the scope.
410	Gone. The requested target is no longer on the recipient and no forwarding URI is known.
411	Size REQUIRED. The requested command MUST be accompanied by byte size or length information in the `Meta` element type.
412	Incomplete command. The requested command failed on the recipient because it was incomplete or incorrectly formed. The recipient SHOULD specify the portion of the command that was incomplete or incorrect in the `Item` element type in the `Status`.
413	Request entity too large. The recipient is refusing to perform the requested command because the requested item is larger than the recipient is able or willing to process. If the condition is temporary, the recipient SHOULD also include a `Status` with the response code 418 and specify a `RespURI` with the response URI and optionally the date/time that the command SHOULD be repeated.
414	URI too long. The requested command failed because the target URI is too long for what the recipient is able or willing to process. This response code is seldom encountered, but is used when a recipient perceives that an intruder might be attempting to exploit security holes or other defects in order to threaten the recipient.
415	Unsupported media type or format. The unsupported content type or format SHOULD also be identified in the `Item` element type in the `Status`.
416	Requested size too big. The request failed because the specified byte size in the request was too big.
417	Retry later. The request failed at this time and the originator SHOULD retry the request later. The recipient SHOULD specify a `RespURI` with the response URI and the date/time that the command SHOULD be repeated.
418	Already exists. The requested `Put` or `Add` command failed because the target already exists.
419	Conflict resolved with server data. The response indicates that the client request created a conflict; which was resolved by the server command winning. The normal information in the `Status` SHOULD be sufficient for the client to "undo" the resolution, if it is desired.
420	Device full. The response indicates that the recipient has no more storage space for the remaining synchronization data. The response includes the remaining number of data that could not be returned to the originator in the `Item` of the `Status`.
421	Unknown filtering grammar. The requested command failed on the server because the specified filtering grammar was not known. The client SHOULD re-specify the filtering using a known filtering grammar.
422	Bad CGI Script. The requested command failed on the server because the CGI scripting was incorrectly formed. The client SHOULD re-specify the portion of the command that was incorrect in the `Item` element type in the `Status`.
423	Soft-delete conflict. The requested command failed because the "Soft Deleted" item was previously "Hard Deleted" on the server.
424	Size mismatch. The chunked object was received, but the size of the received object did not match the size declared within the first chunk.
425	Permission Denied. The requested command failed because the sender does not have adequate access control permissions (ACL) on the recipient.

426	Partial item not accepted. Receiver of status code MAY resend the whole item in next package.
427	Item Not empty. Parent cannot be deleted since it contains children.
428	Move Failed
429	Key Exchange Algorithm not supported. The requested command failed because the recipient does not support the Key Exchange Algorithm. The recipient should return the suitable challenge in the Chal element within the Status element.
430	Data Encryption Algorithm not supported. The requested command failed because the recipient does not support the Data Encryption Algorithm. The recipient should return the suitable challenge in the Chal element within the Status element.
431	Key Length not supported. The requested command failed because the recipient does not support the Symmetric Key Length. The recipient should return the suitable challenge in the Chal element within the Status element.
432	Encryption required. The requested command failed because the recipient requires encryption.
Recipient Exception 5xx	
500	Command failed. The recipient encountered an unexpected condition which prevented it from fulfilling the request
501	Command not implemented. The recipient does not support the command REQUIRED to fulfill the request. This is the appropriate response when the recipient does not recognize the requested command and is not capable of supporting it for any resource.
502	Bad gateway. The recipient, while acting as a gateway or proxy, received an invalid response from the upstream recipient it accessed in attempting to fulfill the request.
503	Service unavailable. The recipient is currently unable to handle the request due to a temporary overloading or maintenance of the recipient. The implication is that this is a temporary condition; which will be alleviated after some delay. The recipient SHOULD specify the URI and date/time after which the originator SHOULD retry in the `RespURI` in the response.
504	Gateway timeout. The recipient, while acting as a gateway or proxy, did not receive a timely response from the upstream recipient specified by the URI (e.g. HTTP, FTP, LDAP) or some other auxiliary recipient (e.g. DNS) it needed to access in attempting to complete the request.
505	DTD Version not supported. The recipient does not support or refuses to support the specified version of SyncML DTD used in the request SyncML Message. The recipient MUST include the versions it does support in the `Item` element type in the `Status`.
506	Processing error. An application error occurred while processing the request. The originator SHOULD retry the request. The `RespURI` can contain the URI and date/time after which the originator can retry the request.
507	`Atomic` failed. The error caused all SyncML commands within an `Atomic` element type to fail.
508	Recovery REQUIRED. An error occurred that necessitates a refresh of the current synchronization state of the client with the server. Client is requested to initiate a recovery sync with the server.
509	Reserved for future use.
510	Data store failure. An error occurred while processing the request. The error is related to a failure in the recipient data store.
511	Server failure. A severe error occurred in the server while processing the request. The originator SHOULD NOT retry the request.
512	Synchronization failed. An application error occurred during the synchronization session. The originator SHOULD restart the synchronization session from the beginning.
513	Protocol Version not supported. The recipient does not support or refuses to support the specified version of the SyncML Synchronization Protocol used in the request SyncML Message. The recipient MUST include the versions it does support in the `Item` element type in the `Status`.
514	Operation cancelled. The SyncML command was not completed successfully, since the operation was already cancelled before processing the command. The originator SHOULD repeat the command in the next session.
516	Atomic roll back failed. Command was inside `Atomic` element and `Atomic` failed. This command was not rolled back successfully. Server SHOULD take action to try to recover client back into original state.
517	Atomic response too large to fit. The response to an atomic command was too large to fit in a single message.
518	Refuse to continue the session. The requested command failed because the recipient refuses to continue the session.

Table 18: Response Status Codes

12.Base Media and Content formats

Content	MIME Content Type	URI	Content Format
Contact	text/x-vcard	http://imc.org/pdi/vcard-21.doc	vCard 2.1
	text/directory; profile=vCard	http://www.ietf.org/rfc/rfc2426.txt	vCard 3.0 (official)
	text/vcard	http://www.ietf.org/rfc/rfc2426.txt	vCard 3.0 (common usage)
Calendar	text/x-vcalendar	http://www.imc.org/pdi/vcal-10.doc	vCalendar 1.0
	text/calendar	http://www.ietf.org/rfc/rfc2445.txt	iCalendar 2.0
Memos	text/plain	http://www.ietf.org/rfc/rfc2046.txt	
Tasks	text/x-vcalendar	http://www.imc.org/pdi/vcal-10.doc	vCalendar 1.0
	text/calendar	http://www.ietf.org/rfc/rfc2445.txt	iCalendar 2.0
Email	message/rfc822	http://www.ietf.org/rfc.html	RFC822
			RFC2822
			RFC2045
	application/vnd.omads-email	http://www.openmobilealliance.org/	XML object
File	application/vnd.omads-file	http://www.openmobilealliance.org/	XML object
Folder	application/vnd.omads-folder	http://www.openmobilealliance.org/	XML object

Table 19: Base Media and Content formats

13.MIME Media Type Registration

The following section is the MIME media type registrations for OMA Data Synchronization specific MIME media types.

13.1 application/vnd.syncml+xml

```
To: ietf-types@iana.org

Subject: Registration of MIME media type application/vnd.syncml+xml

MIME media type name: application

MIME subtype name: vnd.syncml+xml

REQUIED parameters: None

OPTIONAL parameters: charset, synctype, verproto, verdtd. May be specified
in any order in the Content-Type MIME header field.

Content-Type MIME header.

charset Parameter

Purpose: Specifies the character set used to represent the SyncML document.
The default character set for SyncML representation protocol is UTF-8, as
defined in [RFC2279].

Formal Specification: The following ABNF defines the syntax for the
parameter.

chrset-param = ";" "charset" "=" <any IANA registered charset identifier>

synctype Parameter

Purpose: Specifies the data synchronization protocol used by the SyncML
document. If present, the value MUST be the same value as that specified by
the "SyncType" element type in the SyncML MIME content information. There is
no default value.

Formal Specification: The following ABNF defines the syntax for the
parameter.

stype-param = ";" "synctype" "=" text

verproto Parameter

Purpose: Specifies the major/minor revision identifiers for the SyncML
synchronization protocol specification for the workflow of messages with
SyncML MIME content. If present, MUST be the same value as that specified by
the "VerProto" element type in the SyncML MIME content information. If not
present, the default value "1.2" is to be assumed.

Formal Specification: The following ABNF defines the syntax for the
parameter.

verprot-param = ";" "verproto" "=" 1*numeric "." 1*numeric

text = 1*ALPHA

numeric = "0" / "1" / "2" / "3" / "4" / "5" / "6" / "7" / "8"/ "9"
```

verdtd Parameter

Purpose: Specifies the major/minor revision identifiers for the SyncML representation protocol specification that defines the SyncML MIME media type. If present, MUST be the same value as that specified by the "VerDTD" element type in the SyncML MIME content information. If not present, the default value "1.2" is to be assumed.

Formal Specification: The following ABNF defines the syntax for the parameter.

```
verdtd-param = ";" "verdtd" "=" 1*numeric "." 1*numeric

text = 1*ALPHA

numeric = "0" / "1" / "2" / "3" / "4" / "5" / "6" / "7" / "8"/ "9"
```

Encoding considerations: The default character set for the SyncML MIME content type is UTF-8. Transfer of this character set through some MIME systems could require that the content is first character encoded into a 7bit character set with an IETF character encoding mechanism such as Base64, as defined in [RFC2045]

Security considerations:

Authentication: The SyncML MIME content type definition provides for the inclusion of authentication information for the purpose of authenticating the originator and recipient of messages containing the data synchronization content type. The content type definition supports Basic, Base64 userid/password mark-up, MD5 digest challenge and response strings and any other registered authentication credential scheme.

Threats: The SyncML MIME content type definition provides for the inclusion of remote execution commands. Administrators for MIME implementations that support this content type SHOULD take every standard precaution to assure the activation of the originator of SyncML content, as well as take every standard precaution to confirm the validity of the included remote execution command prior to allowing the command to be executed on the targeted recipient's system.

Interoperability considerations: Implementations that have support for the mandatory features of this content type will greatly increase the chances of interoperating with other implementations supporting this content type. Conformance to this content type requires an implementation to support every mandatory feature.

Published specification: URL:http://www.openmobilealliance.org/tech/docs

Applications, which use this media type: This MIME content type is intended for common use by networked data synchronization applications.

Additional information:

Magic number(s): None

File extension(s): XSM

Macintosh File Type Code(s): XSML

Person & email address to contact for further information: admins@syncml.org

Intended usage: COMMON

Author/Change controller: mailto:technical-comments@openmobilealliance.org

13.2 application/vnd.syncml+wbxml

To: ietf-types@iana.org

Subject: Registration of MIME media type application/vnd.syncml+wbxml

MIME media type name: application

MIME subtype name: vnd.syncml+wbxml

REQUIRED parameters: None

OPTIONAL parameters: charset, synctype, verproto, verdtd. May be specified in any order in the Content-Type MIME header field.

Content-Type MIME header.

charset Parameter

Purpose: Specifies the character set used to represent the SyncML document. The default character set for SyncML representation protocol is UTF-8, as defined [RFC2279].

Formal Specification: The following ABNF defines the syntax for the parameter.

chrset-param = ";" "charset" "=" <any IANA registered charset identifier>

synctype Parameter

Purpose: Specifies the data synchronization protocol used by the SyncML document. If present, the value MUST be the same value as that specified by the "SyncType" element type in the SyncML MIME content information. There is no default value.

Formal Specification: The following ABNF defines the syntax for the parameter.

stype-param = ";" "synctype" "=" text

verproto Parameter

Purpose: Specifies the major/minor revision identifiers for the SyncML synchronization protocol specification for the workflow of messages with SyncML MIME content. If present, MUST be the same value as that specified by the "VerProto" element type in the SyncML MIME content information. If not present, the default value "1.2" is to be assumed.

Formal Specification: The following ABNF defines the syntax for the parameter.

verprot-param = ";" "verproto" "=" 1*numeric "." 1*numeric

text = 1*ALPHA

numeric = "0" / "1" / "2" / "3" / "4" / "5" / "6" / "7" / "8"/ "9"

verdtd Parameter

Purpose: Specifies the major/minor revision identifiers for the SyncML representation protocol specification that defines the SyncML MIME media type. If present, MUST be the same value as that specified by the "VerDTD" element type in the SyncML MIME content information. If not present, the default value "1.2" is to be assumed.

Formal Specification: The following ABNF defines the syntax for the parameter.

verdtd-param = ";" "verdtd" "=" 1*numeric "." 1*numeric

text = 1*ALPHA

numeric = "0" / "1" / "2" / "3" / "4" / "5" / "6" / "7" / "8"/ "9"

Encoding considerations: The default character set for the SyncML MIME content type is UTF-8. Transfer of this character set through some MIME systems could require that the content is first character encoded into a 7bit character set with an IETF character encoding mechanism such as Base64, as defined in [RFC2045].

Security considerations:

Authentication: The SyncML MIME content type definition provides for the inclusion of authentication information for the purpose of authenticating the originator and recipient of messages containing the data synchronization content type. The content type definition supports Basic, Base64 userid/password mark-up, MD5 digest challenge and response strings and any other registered authentication credential scheme.

Threats: The SyncML MIME content type definition provides for the inclusion of remote execution commands. Administrators for MIME implementations that support this content type SHOULD take every standard precaution to assure the authentication of the originator of SyncML content, as well as take every standard precaution to confirm the validity of the included remote execution command prior to allowing the command to be executed on the targeted recipient's system.

Interoperability considerations: Implementations that have support for the mandatory features of this content type will greatly increase the chances of interoperating with other implementations supporting this content type. Conformance to this content type requires an implementation to support every mandatory feature.

Published specification:
http://www.syncml.org/docs/syncml_sync_represent_v111_20021002.pdf

Applications, which use this media type: This MIME content type is intended for common use by networked data synchronization applications.

Additional information:

Magic number(s): None

File extension(s): BSM

Macintosh File Type Code(s): BSML

Person & email address to contact for further information: admins@syncml.org

Intended usage: COMMON

Author/Change controller: mailto:technical-comments@openmobilealliance.org

Appendix A. [SYNTAX] Static Conformance Requirements (Normative)

The SCR tables in this Appendix form a profile of the Static Conformance Requirements.

Static conformance requirements (SCR) specify the features that are OPTIONAL and MANDATORY within implementations conforming to this specification.

Further static conformance requirements for the usage of this specification are specified in [DSPRO].

A.1 Client Features

A.1.1 Security

Item	Function	Ref.	Status	Requirement
DS-Syntax-SEC-C-001	Support Basic and SHA-256	6.2.17	M	
DS-Syntax-SEC-C-002	Support optional authentication types	6.2.17	O	

Table 20: Client Features - Security

A.1.2 XML Usage

Item	Function	Ref.	Status	Requirement
DS-Syntax-XML-C-001	Support namespace usage	5.3	M	

Table 21: Client Features - XML Usage

A.1.3 MIME Usage

Item	Function	Ref.	Status	Requirement
DS-Syntax-MIME-C-001	Support MIME content types	DSPRO	M	

Table 22: Client Features - MIME Usage

A.1.4 Identifiers

Item	Function	Ref.	Status	Requirement
DS-Syntax-IDS-C-001	Support URI, URN, textual names	DSPRO	M	

Table 23: Client Features - Identifiers

A.1.5 Common Use Elements

Item	Function	Ref.	Status	Requirement
DS-Syntax-CUE-C-001	Support Anchor	6.2.3	M	
DS-Syntax-CUE-C-002	Support Atomic	6.2.4	O	
DS-Syntax-CUE-C-003	Support AuthName	6.2.5	O	
DS-Syntax-CUE-C-004	Support Behaviour	6.2.6	M	
DS-Syntax-CUE-C-005	Support Chal	6.2.7	M	DS-Syntax-DDE-C-003
DS-Syntax-CUE-C-006	Support ChangeLogValidity	6.2.8	M	
DS-Syntax-CUE-C-007	Support ClientURI	6.2.9	M	
DS-Syntax-CUE-C-008	Support Cmd	6.2.10	M	
DS-Syntax-CUE-C-009	Support CmdID	6.2.11	M	
DS-Syntax-CUE-C-010	Support CmdRef	6.2.12	M	
DS-Syntax-CUE-C-011	Support Code	6.2.13	M	
DS-Syntax-CUE-C-012	Support Correlator	6.2.16	O	
DS-Syntax-CUE-C-013	Support Cred	6.2.17	M	
DS-Syntax-CUE-C-014	Support Direction	6.2.20	M	
DS-Syntax-CUE-C-015	Support Encrypted	6.2.22	O	
DS-Syntax-CUE-C-016	Support EncryptedKey	6.2.23	O	

Item	Function	Ref.	Status	Requirement
DS-Syntax-CUE-C-017	Support Field	6.2.24	O	
DS-Syntax-CUE-C-018	Support FieldLevel	6.2.25	O	
DS-Syntax-CUE-C-019	Support Filter	6.2.26	O	DS-Syntax-DDE-C-003
DS-Syntax-CUE-C-020	Support Filter Type	6.2.27	O	
DS-Syntax-CUE-C-021	Support Final	6.2.28	M	
DS-Syntax-CUE-C-022	Support Format	6.2.29	O	
DS-Syntax-CUE-C-023	Support FP	6.2.30	M	
DS-Syntax-CUE-C-024	Support FreeID	6.2.31	O	
DS-Syntax-CUE-C-025	Support FreeMem	6.2.32	O	
DS-Syntax-CUE-C-026	Support ID	6.2.34	O	
DS-Syntax-CUE-C-027	Support IDContainer	6.2.35	O	
DS-Syntax-CUE-C-028	Support IDValidity	6.2.36	M	
DS-Syntax-CUE-C-029	Support Last	6.2.38	M	
DS-Syntax-CUE-C-030	Support MaxMsgSize	6.2.39	O	
DS-Syntax-CUE-C-031	Support MaxObjSize	6.2.40	O	
DS-Syntax-CUE-C-032	Support MoreData	6.2.42	O	
DS-Syntax-CUE-C-033	Support MsgID	6.2.44	M	
DS-Syntax-CUE-C-034	Support MsgRef	6.2.45	M	
DS-Syntax-CUE-C-035	Support Next	6.2.46	M	
DS-Syntax-CUE-C-036	Support NextNonce	6.2.47	M	
DS-Syntax-CUE-C-037	Support NoStatus	6.2.48	M	
DS-Syntax-CUE-C-038	Support NumberOfChanges	6.2.49	O	
DS-Syntax-CUE-C-039	Support Record	6.2.51	O	
DS-Syntax-CUE-C-040	Support RespURI	6.2.53	O	
DS-Syntax-CUE-C-041	Support Sequence	6.2.55	O	
DS-Syntax-CUE-C-042	Support ServerURI	6.2.56	M	
DS-Syntax-CUE-C-043	Support SessionID	6.2.57	M	
DS-Syntax-CUE-C-044	Support SftDel	6.2.58	O	
DS-Syntax-CUE-C-045	Support Size	6.2.59	M	
DS-Syntax-CUE-C-046	Support SourceClientURI	6.2.60	M	
DS-Syntax-CUE-C-047	Support SourceClientParentURI	6.2.61	O	
DS-Syntax-CUE-C-048	Support SourceServerURI	6.2.62	M	
DS-Syntax-CUE-C-049	Support SourceServerParentURI	6.2.63	O	
DS-Syntax-CUE-C-050	Support StatusItem	6.2.65	O	
DS-Syntax-CUE-C-051	Support SyncType	6.2.71	M	
DS-Syntax-CUE-C-052	Support TargetClientURI	6.2.72	M	
DS-Syntax-CUE-C-053	Support TargetClientParentURI	6.2.73	O	
DS-Syntax-CUE-C-054	Support TargetServerURI	6.2.74	M	
DS-Syntax-CUE-C-055	Support TargetServerParentURI	6.2.75	O	
DS-Syntax-CUE-C-056	Support Type	6.2.76	M	
DS-Syntax-CUE-C-057	Support Version	6.2.77	M	

Table 24: Client Features - Common Use Elements

A.1.6 Message Container Elements

Item	Function	Ref.	Status	Requirement
DS-Syntax-MCE-C-001	Support SyncML	6.2.70	M	DS-Syntax-MCE-C-002 AND DS-Syntax-MCE-C-003
DS-Syntax-MCE-C-002	Support SyncHdr	6.2.69	M	
DS-Syntax-MCE-C-003	Support SyncBody	6.2.68	M	

Table 25: Client Features - Message Container Elements

A.1.7 Data Description Elements

Item	Function	Ref.	Status	Requirement

Item	Function	Ref.	Status	Requirement
DS-Syntax-DDE-C-001	Support Data	6.2.18	M	
DS-Syntax-DDE-C-002	Support Item	6.2.37	M	
DS-Syntax-DDE-C-003	Support Meta	6.2.41	M	

Table 26: Client Features - Data Description Elements

A.1.8 Protocol Management Elements

Item	Function	Ref.	Status	Requirement
DS-Syntax-PME-C-001	Support Status	6.2.64	M	

Table 27: Client Features - Protocol Management Elements

A.1.9 Protocol Command Elements

Item	Function	Ref.	Status	Requirement
DS-Syntax-PCE-C-001	Support Add	6.2.1	M	
DS-Syntax-PCE-C-002	Support Alert	6.2.2	M	
DS-Syntax-PCE-C-003	Support Copy	6.2.15	O / M	Mandatory if Hierarchical Sync is supported
DS-Syntax-PCE-C-004	Support Delete	6.2.19	M	
DS-Syntax-PCE-C-005	Support SyncAlert	6.2.67	M	
DS-Syntax-PCE-C-006	Support Get	6.2.33	M	
DS-Syntax-PCE-C-007	Support Move	6.2.43	O / M	Mandatory if Hierarchical Sync is supported
DS-Syntax-PCE-C-008	Support Put	6.2.50	M	
DS-Syntax-PCE-C-009	Support Replace	6.2.52	M	
DS-Syntax-PCE-C-010	Support Results	6.2.54	O	
DS-Syntax-PCE-C-011	Support Sync	6.2.65	M	

Table 28: Client Features - Protocol Command Elements

A.2 Server Features

A.2.1 Security

Item	Function	Ref.	Status	Requirement
DS-Syntax-SEC-S-001	Support Basic and SHA-256	6.2.17	M	
DS-Syntax-SEC-S-002	Support optional authentication types	6.2.17	O	

Table 29: Server Features - Security

A.2.2 XML Usage

Item	Function	Ref.	Status	Requirement
DS-Syntax-XML-S-001	Support namespace usage	5.3	M	

Table 30: Server Features - XML Usage

A.2.3 MIME Usage

Item	Function	Ref.	Status	Requirement
DS-Syntax-MIME-S-001	Support MIME content types	DSPRO	M	

Table 31: Server Features - MIME Usage

A.2.4 Identifiers

Item	Function	Ref.	Status	Requirement
DS-Syntax-IDS-S-001	Support URI, URN, textual names	DSPRO	M	

Table 32: Server Features - Identifiers

A.2.5 Common Use Elements

Item	Function	Ref.	Status	Requirement
DS-Syntax-CUE-S-001	Support Anchor	6.2.3	M	
DS-Syntax-CUE-S-002	Support Atomic	6.2.4	O	
DS-Syntax-CUE-S-003	Support AuthName	6.2.5	O	
DS-Syntax-CUE-S-004	Support Behaviour	6.2.6	M	
DS-Syntax-CUE-S-005	Support Chal	6.2.7	M	DS-Syntax-DDE-C-003
DS-Syntax-CUE-S-006	Support ChangeLogValidity	6.2.8	M	
DS-Syntax-CUE-S-007	Support ClientURI	6.2.9	M	
DS-Syntax-CUE-S-008	Support Cmd	6.2.10	M	
DS-Syntax-CUE-S-009	Support CmdID	6.2.11	M	
DS-Syntax-CUE-S-010	Support CmdRef	6.2.12	M	
DS-Syntax-CUE-S-011	Support Code	6.2.13	M	
DS-Syntax-CUE-S-012	Support Correlator	6.2.16	O	
DS-Syntax-CUE-S-013	Support Cred	6.2.17	M	
DS-Syntax-CUE-S-014	Support Direction	6.2.20	M	
DS-Syntax-CUE-S-015	Support Encrypted	6.2.22	O	
DS-Syntax-CUE-S-016	Support EncryptedKey	6.2.23	O	
DS-Syntax-CUE-S-017	Support Field	6.2.24	O	
DS-Syntax-CUE-S-018	Support FieldLevel	6.2.25	O	
DS-Syntax-CUE-S-019	Support Filter	6.2.26	O	DS-Syntax-DDE-C-003
DS-Syntax-CUE-S-020	Support Filter Type	6.2.27	O	
DS-Syntax-CUE-S-021	Support Final	6.2.28	M	
DS-Syntax-CUE-S-022	Support Format	6.2.29	O	
DS-Syntax-CUE-S-023	Support FP	6.2.30	M	
DS-Syntax-CUE-S-024	Support FreeID	6.2.31	O	
DS-Syntax-CUE-S-025	Support FreeMem	6.2.32	O	
DS-Syntax-CUE-S-026	Support ID	6.2.34	O	
DS-Syntax-CUE-S-027	Support IDContainer	6.2.35	O	
DS-Syntax-CUE-S-028	Support IDValidity	6.2.36	M	
DS-Syntax-CUE-S-029	Support Last	6.2.38	M	
DS-Syntax-CUE-S-030	Support MaxMsgSize	6.2.39	O	
DS-Syntax-CUE-S-031	Support MaxObjSize	6.2.40	O	
DS-Syntax-CUE-S-032	Support MoreData	6.2.42	M	
DS-Syntax-CUE-S-033	Support MsgID	6.2.44	M	
DS-Syntax-CUE-S-034	Support MsgRef	6.2.45	M	
DS-Syntax-CUE-S-035	Support Next	6.2.46	M	
DS-Syntax-CUE-S-036	Support NextNonce	6.2.47	M	
DS-Syntax-CUE-S-037	Support NoStatus	6.2.48	M	
DS-Syntax-CUE-S-038	Support NumberOfChanges	6.2.49	M	
DS-Syntax-CUE-S-039	Support Record	6.2.51	O	
DS-Syntax-CUE-S-040	Support RespURI	6.2.53	M	
DS-Syntax-CUE-S-041	Support Sequence	6.2.55	O	
DS-Syntax-CUE-S-042	Support ServerURI	6.2.56	M	
DS-Syntax-CUE-S-043	Support SessionID	6.2.57	M	
DS-Syntax-CUE-S-044	Support SftDel	6.2.58	O	
DS-Syntax-CUE-S-045	Support Size	6.2.59	M	

Item	Function	Ref.	Status	Requirement
DS-Syntax-CUE-S-046	Support SourceClientURI	6.2.60	M	
DS-Syntax-CUE-S-047	Support SourceClientParentURI	6.2.61	O	
DS-Syntax-CUE-S-048	Support SourceServerURI	6.2.62	M	
DS-Syntax-CUE-S-049	Support SourceServerParentURI	6.2.63	O	
DS-Syntax-CUE-S-050	Support StatusItem	6.2.65	O	
DS-Syntax-CUE-S-051	Support SyncType	6.2.71	M	
DS-Syntax-CUE-S-052	Support TargetClientURI	6.2.72	M	
DS-Syntax-CUE-S-053	Support TargetClientParentURI	6.2.73	O	
DS-Syntax-CUE-S-054	Support TargetServerURI	6.2.74	M	
DS-Syntax-CUE-S-055	Support TargetServerParentURI	6.2.75	O	
DS-Syntax-CUE-S-056	Support Type	6.2.76	M	
DS-Syntax-CUE-S-057	Support Version	6.2.77	M	

Table 33: Server Features - Common Use Elements

A.2.6 Message Container Elements

Item	Function	Ref.	Status	Requirement
DS-Syntax-MCE-S-001	Support SyncML	6.2.70	M	
DS-Syntax-MCE-S-002	Support SyncHdr	6.2.69	M	
DS-Syntax-MCE-S-003	Support SyncBody	6.2.68	M	

Table 34: Server Features - Message Container Elements

A.2.7 Data Description Elements

Item	Function	Ref.	Status	Requirement
DS-Syntax-DDE-S-001	Support Data	6.2.18	M	
DS-Syntax-DDE-S-002	Support Item	6.2.37	M	
DS-Syntax-DDE-S-003	Support Meta	6.2.41	M	

Table 35: Server Features - Data Description Elements

A.2.8 Protocol Management Elements

Item	Function	Ref.	Status	Requirement
DS-Syntax-PME-S-001	Support Status	6.2.64	M	

Table 36: Server Features - Protocol Management Elements

A.2.9 Protocol Command Elements

Item	Function	Ref.	Status	Requirement
DS-Syntax-PCE-S-001	Support Add	6.2.1	M	
DS-Syntax-PCE-S-002	Support Alert	6.2.2	M	
DS-Syntax-PCE-S-003	Support Copy	6.2.15	O / M	Mandatory if Hierarchical Sync is supported
DS-Syntax-PCE-S-004	Support Delete	6.2.19	O	
DS-Syntax-PCE-S-005	Support Get	6.2.33	O	
DS-Syntax-PCE-S-006	Support Move	6.2.43	O / M	Mandatory if Hierarchical Sync is supported
DS-Syntax-PCE-S-007	Support Put	6.2.50	M	
DS-Syntax-PCE-S-008	Support Replace	6.2.52	M	
DS-Syntax-PCE-S-009	Support Results	6.2.54	M	

Item	Function	Ref.	Status	Requirement
DS-Syntax-PCE-S-010	Support SyncAlert	6.2.55	M	
DS-Syntax-PCE-S-011	Support Sync	6.2.65	M	

Table 37: Server Features - Protocol Command Elements

Appendix B. Example Validation Aids (Informative)

Many of the examples of this document may be validated against the schema. Since the examples are generally incomplete, they need to be incorporated into an appropriate framework for this to work. Some of the frameworks used, and the examples that can validate with them are shown below.

B.1 Example Data Hierarchy

Figure 1 contains an example hierarchy of objects. This collection is either the starting point or the ending point for many of the examples.

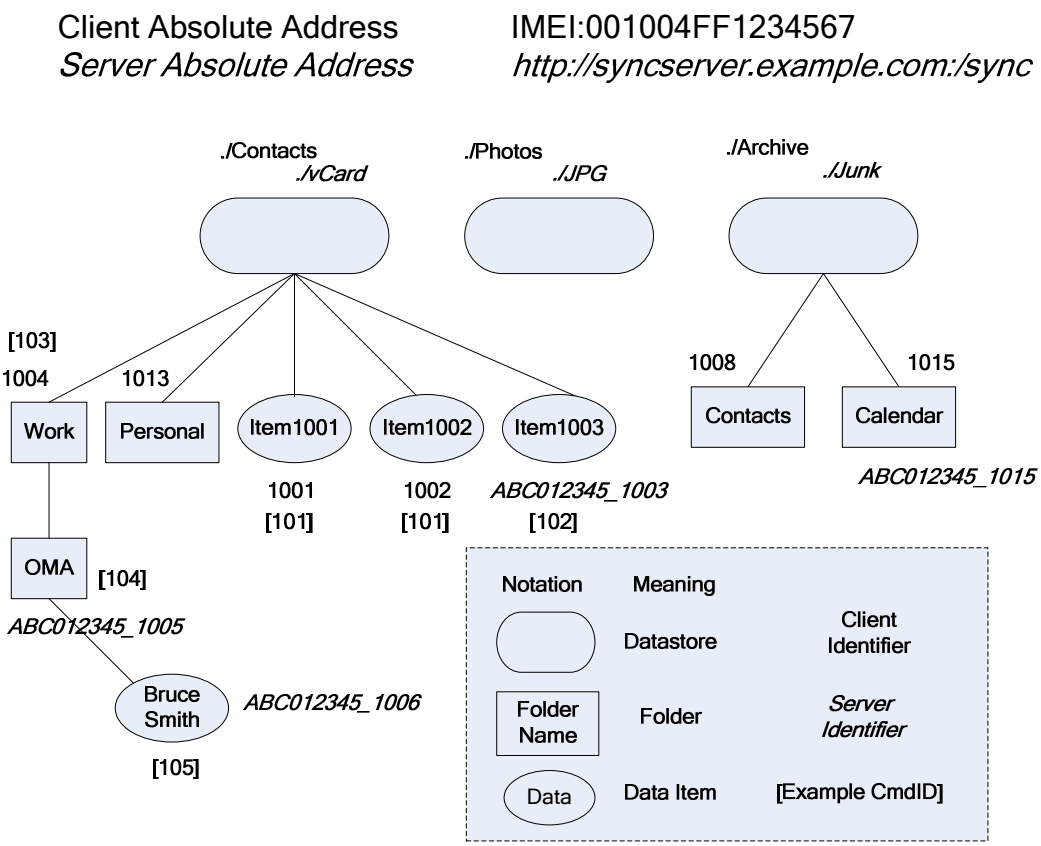

Figure 1 - Example of Hierarchy

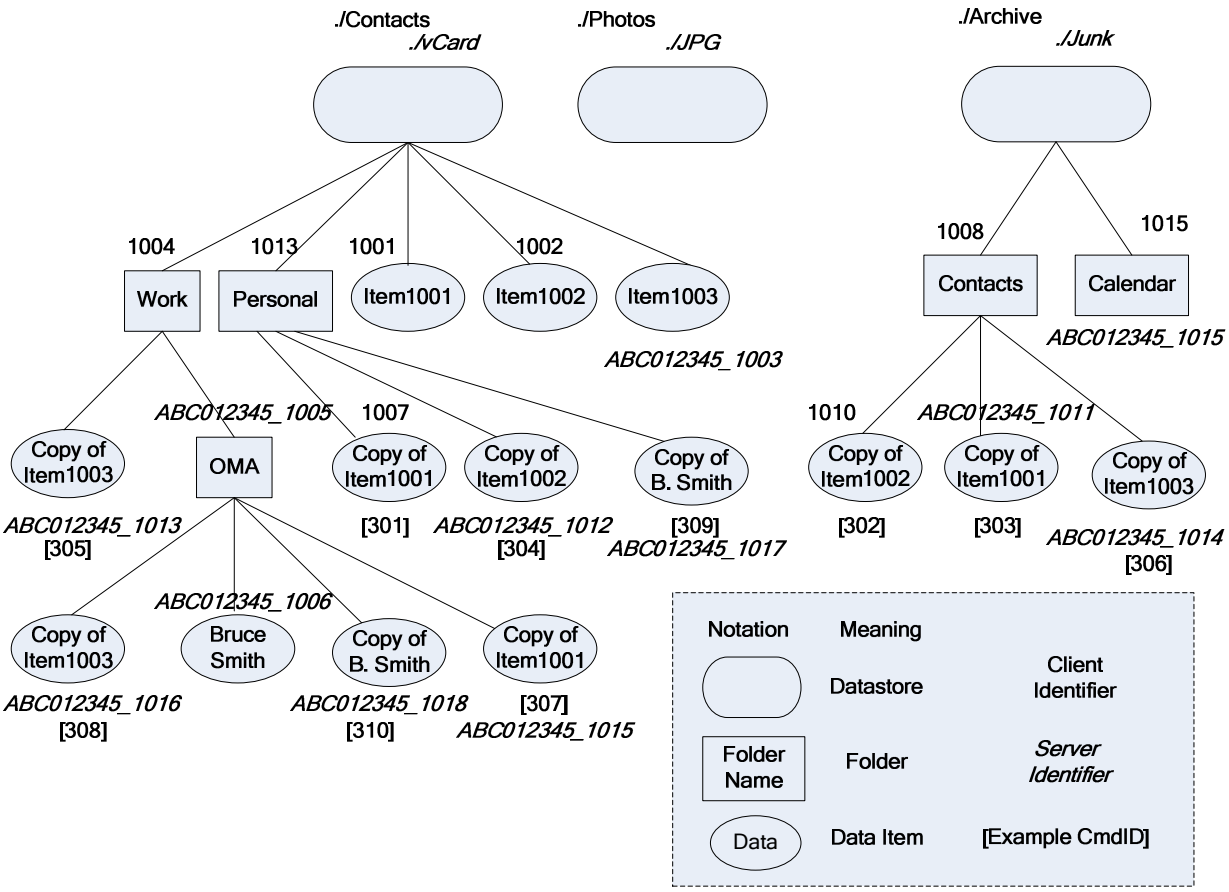

Figure 2 - Example of Modified Hierarchy

Figure 2 shows the hierarchy as modified during the `Copy` and `Move` commands.

B.2 Inside Sync Element Examples

Examples for the following commands may be validated with this framework: `Add`, `Del`

```
<?xml version="1.0" encoding="UTF-8"?>
<SyncML xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation=
  "OMA-SUP-XSD_DS_Syntax_Schema-V2_0.xsd" Version="2.0">

  <SyncHdr MsgID="1" SessionID="100">
    <TargetServerURI>ServerURIGoesHere</TargetServerURI>
    <SourceClientURI>ClientIdentifierGoesHere</SourceClientURI>
  </SyncHdr>
  <SyncBody>
```

```
    <Sync CmdID="1">
      <!-- Insert Example Below Here -->

      <!-- Insert Example Above Here -->
    </Sync>
  </SyncBody>
</SyncML>
```

Appendix C. Change History (Informative)

C.1 Approved Version 2.0 History

Reference	Date	Description
OMA-TS-DS_Syntax-V2_0- 20110719-A	19 Jul 2011	Status changed to Approved by TP: OMA-TP-2011-0258-INP_DS_V2_0_ERP_for_final_Approval